Apple Assembly Line

$2.40

Volume 8 -- Issue 4 January, 1988

New Subscription Rates

For the first time since January of 1984, we are going to have to increase our subscription rates. The Post Office is raising the postage again, for the third time since then, and we have to respond. Of course postage is not the only expense that has increased, just the most recent, and most noticeable. Here are the old and new rates for a year's subscription:

	Current	New
Newsletter Only, 12 issues:		
Bulk Mail (USA only)	$18	---
First Class Mail (USA, Canada, Mexico)	$21	$24
All Other Countries	$32	$36
Newsletter plus the Monthly Disk, 12 issues:		
USA, Canada, Mexico	$64	$64
All Other Countries	$87	$90

You will notice that the Bulk Mail option is being phased out, as I am planning to mail by First Class to all USA subscribers. The reliability of Bulk Mail has been entirely too erratic, and increasingly so.

You will also notice that the price of a subscription including disk, delivered in the USA, Canada, or Mexico, has not increased at all. Maybe now is the time to upgrade, and save yourself a lot of typing?

The new prices go into effect for new subscriptions as of March 15, 1988. Renewals at the old prices (but no more bulk mail) will continue to be accepted through the 4th of April.

Another Peek Inside AppleWorks: The Interpretive String Display Subroutine...Bob Sander-Cederlof

At least twice in the last eight years I have published fancy message display subroutines. In the original Volume 1, Number 1, way back in October, 1980, I gave a really nice 40-column version. That one was actually used in a slightly different form inside a system that we developed for the American Heart Association for teaching Cardio-Pulmonary Resuscitation (CPR).

In the April 1987 issue I published an 80-column version for the //e and //c, which had similar but more powerful capabilities.

Last month I started revealing some of the code from inside AppleWorks (version 1.3). I covered parameter passing and some string handling subroutines, plus block move. I also described and printed the subroutine which polls the keyboard, so that you can type before being asked. POLL.KEYBOARD is called from all over everywhere, just to be sure no characters are lost. I mention that, because there is a call to it from the code I am unveiling this month. In the listing which follows, I have put a dummy POLL.KEYBOARD subroutine which simply does an RTS. In the real AppleWorks code, DISPLAY.STRING calls the real POLL.KEYBOARD.

As I mentioned last month, I am NOT showing in these pages the exact code you would find inside AppleWorks. Since my purpose is not to document AppleWorks, but rather to cull out generally useful code which we can adapt and use, I have rearranged and modified a little. My versions are, in general, shorter and faster. Maybe whoever is currently maintaining AppleWorks at Claris will notice and use these improvements.

In case you ARE interested in documenting AppleWorks, or just want to see what I changed, I have included comment lines with each segment of code which show what the corresponding address was inside AppleWorks 1.3. As was true last month, all the code shown here is found in the main segment called APLWORKS.SYSTEM, which begins at $1000 once it is up and running.

I will begin with a general description of features. DISPLAY.STRING began at $14D1, and there are numerous calls to it in the code. There is also a JMP vector to it in the JMP table which begins at $1000; if you find any JSR $1015 instructions in any of the other segments, they are calling this DISPLAY.STRING subroutine. Unlike all of the subroutines I discussed last month, this subroutine does not expect to find parameters following the JSR which called it. Instead, it expects the length of the string to be in the A-register, and the address of the string to be in locations $80,81. (There is another subroutine which uses the parameter-passing protocol to display a string which starts with a length byte; it simply sets up $80, $81, and the A-register and calls DISPLAY.STRING. You can find it at $2093, with a JMP vector at $1087.)

S-C Macro Assembler Version 2.0	DOS $100, ProDOS $100, both for	$120	
Version 2.0 DOS Upgrade Kit for 1.0/1.1/1.2 owners		$20	
ProDOS Upgrade Kit for Version 2.0 DOS owners		$30	
Source Code of S-C Macro 2.0 (DOS or ProDOS)	each, additional	$100	
S-C DisAssembler (ProDOS only)	without source code	$30, with source	$50
RAK-Ware DISASM (DOS only)	without source code	$30, with source	$50
ProVIEW (ProDOS only)		$20	
Full Screen Editor for S-C Macro (with complete source code)		$49	
S-C Cross Reference Utility	without source code	$20, with source	$50
S-C Word Processor (with complete source code)		$50	
DP18 and DPFP, including complete source and object code		$50	
ES-CAPE (Extended S-C Applesoft Program Editor), including Version 2.0 With Source Code		$50	
ES-CAPE Version 2.0 and Source Code Update (for Registered Owners)		$30	
Bag of Tricks 2 (Quality Software)	($49.95)	$45	*
S-C Documentor (complete commented source code of Applesoft ROMs)		$50	
Cross Assemblers for owners of S-C Macro Assembler		$32.50 to $50 each	
(Available: 6800/1/2, 6301, 6805, 6809, 68000, Z-80, Z-8, 8048, 8051, 8085, 1802/4/5, PDP-11, GI1650/70, Mitsubishi 50740, others)			
Beagle Bros. Applesoft Compiler (ProDOS only)	($74.95)	$65	*
Copy II Plus (Central Point Software)	($39.95)	$30	*
AAL Quarterly Disks	each $15, or any four for	$45	

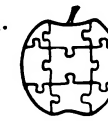
(All source code is formatted for S-C Macro Assembler. Other assemblers require some effort to convert file type and edit directives.)

Vinyl disk pages, 6"x8.5", hold two disks each	10 for	$6	*
Diskette Mailing Protectors (hold 1 or 2 disks)	40 cents each		
(Cardboard folders designed to fit 6"X9" Envelopes.)	or $25 per 100		*
Envelopes for Diskette Mailers	6 cents each		
Sider 20 Meg Hard Disk, includes controller & software	($695)	$550	+
Sider 40 Meg Hard Disk, includes controller & software	($995)	$860	+
Minuteman 250 Uninterruptible Power Supply	($359)	$320	+
Minuteman 300 Uninterruptible Power Supply	($549)	$490	+
65802 Microprocessor, 4 MHz (Western Design Center)		$25	*
quikLoader EPROM System (SCRG)	($179)	$170	*
PROmGRAMER (SCRG)	($149.50)	$140	*

"Exploring the Apple IIgs"	Gary B. Little	($22.95)	$21 *
"Apple IIgs Technical Reference"	Michael Fischer	($19.95)	$19 *
"65816/65802 Assembly Language Programming"	Michael Fischer	($21.95)	$20 *
"Programming the 65816"	David Eyes & Ron Lichty	($22.95)	$21 *
"Apple //e Reference Manual"	Apple Computer	($24.95)	$23 *
"Apple //c Reference Manual"	Apple Computer	($24.95)	$23 *
"ProDOS-8 Technical Reference Manual"	Apple Computer	($29.95)	$27 *
"ProDOS-16 Technical Reference Manual"	Apple Computer	($29.95)	$27 *
"Apple IIgs Firmware Reference"	Apple Computer	($24.95)	$23 *
"Apple IIgs Hardware Reference"	Apple Computer	($24.95)	$23 *
"ProDOS Inside and Out"	Dennis Doms & Tom Weishaar	($16.95)	$16 *
"DOStalk Scrapbook"	Tom Weishaar & Bert Kersey	($14.95)	$14 *
"Beneath Apple ProDOS"	Don Worth & Pieter Lechner	($19.95)	$18 *
"Beneath Apple DOS"	Don Worth & Pieter Lechner	($19.95)	$18 *
"Inside the Apple //c"	Gary B. Little	($19.95)	$18 *
"Inside the Apple //e"	Gary B. Little	($19.95)	$18 *
"Understanding the Apple //e"	Jim Sather	($24.95)	$23 *
"Understanding the Apple II"	Jim Sather	($22.95)	$21 *
"Apple II+/IIe Troubleshooting & Repair Guide"	Brenner	($19.95)	$18 *
"Assembly Language for Applesoft Programmers"	Finley & Myers	($18.95)	$18 *
"Now That You Know Apple Assembly Language"	Jules Gilder	($19.95)	$18 *
"Enhancing Your Apple II, vol. 1"	Don Lancaster	($15.95)	$15 *
"Enhancing Your Apple II, vol. 2"	Don Lancaster	($17.95)	$17 *
"Assembly Cookbook for the Apple II/IIe"	Don Lancaster	($21.95)	$20 *
"Microcomputer Graphics"	Roy E.Myers	($14.95)	$14 *
"Assembly Lines -- the Book"	Roger Wagner	($19.95)	$12 *

* These items add $2 for first item, $.75 for each additional item for US shipping.
\+ Inquire for shipping cost.
Customers outside USA inquire for postage needed.
Texas residents please add 8% sales tax to all orders.
<< Master Card, VISA, Discover and American Express >>

S-C Software Corporation
2331 Gus Thomasson #125
DALLAS, TX 75228
Phone 214-324-2050



DISPLAY.STRING does not use any Apple firmware at all. The display techniques used here work faster than the firmware, because they are dedicated to 80-columns and do not have to retain any compatibility with older machines. If you remember that the original Apple //e firmware scrolled the 80-column screen with a slow zigzag motion, you can see why Rupert Lissner decided to code his own.

The characters within the string to be displayed consist of control codes and displayable characters. Displayable characters include the full upper- and lower-case alphabet, numbers, and punctuation signs; all of these can be displayed in both normal and inverse mode. All 32 "mouse-text" characters can also be displayed, although the only one I have noticed in quickly scanning through the AppleWorks messages is the open-apple.

Inverse and normal mode is controlled by a flag byte, which I have called INVERSE.FLAG in my code. It is located at $14D0 in AppleWorks. If that byte contains $00, characters will display in inverse; if $80, normal. A pair of control codes lets you switch INVERSE.FLAG back and forth from within a string, or you can directly set it between calls to DISPLAY.STRING.

The following table shows the hex values for the various character groups as interpreted by DISPLAY.STRING:

```
00-1F  Control Codes
20-7F  96 ASCII Characters
80-9F  32 Mouse Text Characters
```

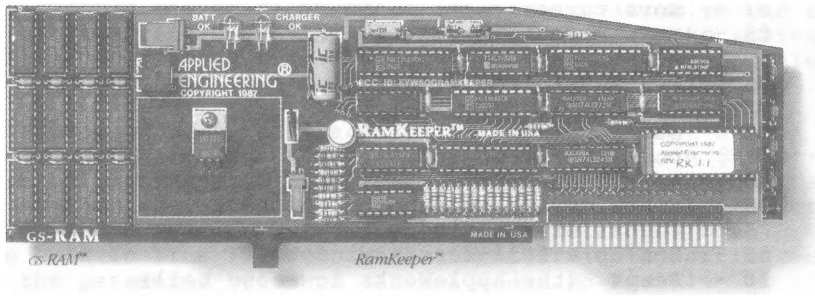
The codes 20-7F display in either normal or inverse, depending on INVERSE.FLAG, as described above. Codes C0-DF duplicate 80-9F, displaying the mouse text characters; codes A0-BF and E0-FF both display the 32 lower-case characters in inverse mode.

There are a couple of un-features in DISPLAY.STRING. There is a JMP $1815 instruction located at $1815 inside AppleWorks 1.3. This, of course, hangs up the Apple. The only way out is by hitting RESET. DISPLAY.STRING goes to this HANG.UP code under some circumstances. Since it is a deadly trap, I assume the author of AppleWorks used to have some debugging code there. It gets called from all over, when errors occur that are programming bugs. There is even a JMP vector to it in the JMP table, at $101B! I have left the jumps and branches to HANG.UP in my version, but you might want to modify them to do something reasonable if you are going to use DISPLAY.STRING yourself.

The other un-feature is what happens if you try to print past the right edge of the text window. You might think it would automatically wrap to the next line, like the standard Apple firmware does; no, it just piles up the characters at the end of the line like old manual typewriters used to do. Possibly you might consider a third un-feature to be the limitation of 255 characters in a string, but this is easy enough to work around. My demonstration program, included at the end of the following listing, shows one way.

RamKeeper™

For the "Instant On" Apple IIGS.



Permanent Storage with an "Electronic Hard Disk"

Now when you turn on your IIGS your favorite program can appear on screen in just a few seconds! With RamKeeper, your IIGS memory card will retain stored programs and stored data while your IIGS is turned off. RamKeeper allows you to divide your IIGS memory into part "electronic hard disk" and part RAM for your programs workspace—in almost any way you want and at anytime you want. GS-RAM, GS-RAM Plus, Apple IIGS memory card and most other IIGS memory cards are compatible with RamKeeper.

Supports Up to Two IIGS Memory Cards at the Same Time

If you bought your IIGS with Apple's memory card and later wished you had the GS-RAM, no problem. RamKeeper will support both cards plugged into RamKeeper simultaneously!

How it Works

Just unplug your IIGS memory card from your computer, plug your IIGS memory card into RamKeeper, plug RamKeeper into the IIGS memory slot. If you have another IIGS memory card, an additional card socket on RamKeeper will accommodate your second card. That's all there is to it!

"I've purchased several Applied Engineering products over the years. They're always well made and performed as advertised. I recommend them whole-heartedly."

Steve Wozniak, the creator of Apple Computer

Reliability from the Most Experienced

Applied Engineering has the most experience in the industry with battery-backed memory for the Apple so you are assured of the most reliable memory back-up system available. And in the world of battery-backed memory, reliability is everything! That's why Applied Engineering uses the more dependable Gel-Cell's instead of Ni-Cad batteries (if Ni-Cad's aren't discharged periodically, they lose much of their capacity). RamKeeper has close to 6 times (about 6 hours) the "total power failure" back-up time of other systems. When power returns, RamKeeper automatically re-charges the battery to a full charge. With power from your wall outlet, RamKeeper will back-up your IIGS memory cards RAM indefinately.

RamKeeper Has and Does It All!

- Allows instant access to your programs without slow disk delays
- Configure Kilobytes or Megabytes of instant ROM storage for your favorite programs
- Reduces power strain to your internal IIGS' power supply
- Contains back-up status L.E.D.'s
- Can support up to two IIGS memory cards simultaneously
- Supports both 256K-installed memory chip boards like GS-RAM and the Apple IIGS Memory Expansion Card as well as 1 MEG installed memory chip boards like GS-RAM Plus
- 5-year hassle-free warranty
- 15 day money back guarantee
- Proudly made in the U.S.A.
- RamKeeper comes *complete* with battery, software and documentation

Only $179.00!

(GS-RAM card shown in photo not included)

Order Your RamKeeper Today!

See your dealer or call (214) 241-6060, 9:00-11:00 CST, 7 days a week, or send check or money order to Applied Engineering. MasterCard, VISA and C.O.D. welcome. Texas residents add 7% sales tax. Add $10.00 if outside U.S.A.

APPLIED ENGINEERING™

The Apple enhancement experts.

(214) 241-6060

P.O. Box 798, Carrollton, TX 75006

Prices subject to change without notice.

There are 17 control codes interpreted by DISPLAY.STRING, and room for adding 15 more. Most of these are single byte codes, but two are followed by parameter bytes. Here is a table of the codes:

```
      01 -- Clear from cursor to end of line.
      02 -- Clear entire line cursor is on.
      03 -- Home (go to 0,0 and clear window).
      04 -- Clear from cursor to end of window.
  05xxyy -- Move cursor to column xx, row yy of window
            (HANG.UP if string ends with no xxyy).
      06 -- Move cursor left (HANG.UP if beyond window).
      07 -- Move cursor right (HANG.UP if beyond window).
      08 -- Move cursor up, scroll if already at top.
      09 -- Move cursor down, scroll if alread at bottom.
      0A -- Set inverse mode.
      0B -- Set normal mode.
      0C -- Store current cursor position as bottom-right
            corner of the window.
      0D -- Move cursor to beginning of current line.
      0E -- Store current cursor position as top-left
            corner of the window.
      0F -- Set up a full-screen window.
      10 -- Beep!  (the AppleWorks low-tone bell).
    11xx -- Slide the screen sideways xx columns.
            If xx>0 slide to right; if xx<0, slide left.
```

You can see that setting windows is fairly easy. Use code 05xxyy to position the cursor where you want the bottom-right corner of the new window to be, and then use code 0C to store it; then use 05xxyy to position the cursor where you want the top-left corner of the new window to be, and then use code 0E to make the new window. Since all 05xxyy moves are relative to the current window, you need to set the new bottom-right corner first. (You should now have a clue how AppleWorks nests the file folders on the screen.)

Since AppleWorks does not use any of the Apple firmware, it is also not tied to the standard page-zero locations for window info and cursor position. DISPLAY.STRING uses $10 through $13 to store the window definition. It is not the same as Apple's window definition bytes at $20-$23. Apple's firmware uses a starting column and a width, whereas AppleWorks uses a starting column and an ending column. The bytes are in a different order, too. See lines 1030-1130 for DISPLAY.STRINGs page zero-usage. The two bytes defined in lines 1120-1130, at $F0 and $F1, are the ones AppleWorks uses. However, you could change those two lines to share $18 and $19 with the labels defined on lines 1090-1100, if you wish.

Lines 1150-1320 define two macros, one for storing a byte on the screen and one for picking a byte off the screen. There is another macro definition in lines 4820-4840, for the function vector table. I decided to put these in as macros to shorten the program listing so it would fit in this issue of AAL. It also makes the code in the left-right scroll subroutine easier to follow. My code inside these macros is different from the

code in AppleWorks: it is shorter, and on the average one cycle SLOWER. I more than made up for the speed loss in other places, though.

You will find the main body of DISPLAY.STRING in lines 1390-1920. Lines 1470-1480 are curious. They cause the entire call to DISPLAY.STRING to be ignored if the contents of $A4 is non-zero. I don't know why or when AppleWorks would use this. Probably you will want to delete these two lines if you use a revision of DISPLAY.STRING in your own programs. In that case, you would want to substitute a LDA #0 instruction, so that line 1490 would store a zero as the beginning position in the string to be displayed. Line 1500 calls the POLL.KEYBOARD subroutine, which as I mentioned above is just a dummy routine in this listing. You will probably want to delete this line too.

By the time we get to line 1520, everything is set up. A pointer to the first character of the string is in $80,81; POSITION.IN.STRING holds the index relative to that pointer for the next character to be processed; INVERSE.FLAG is either 00 or 80; and BYTES.IN.STRING is set to the string length. We come back to line 1520 for each character in the string, except for the parameter bytes on control codes 05 and 11.

Lines 1520-1540 test to see if we have finished the string, and go to an RTS if so. Lines 1550-1670 pick up the next character, and decide how to process it according to the range: values 80-FF are treated as mouse text, even though we only expect 80-9F for these; values 00-1F are control (function) codes; and values 20-7F are regular ASCII characters.

Mouse text characters are really put on the screen by using values from $40 to $5F, so lines 1680-1700 do the honors.

Regular ASCII characters are EORed with the INVERSE.FLAG in line 1600. If the result is negative, we have a "normal" character; if positive, "inverse". Normal characters are ready now to display, but inverse take some care. The range 40-5F should print as letters rather than mouse text, so they are mapped down to 00-1F in lines 1620-1670.

Control codes are handled in lines 1830-1920. This is not the same way AppleWorks did it. AppleWorks used the trick of pushing the table address on the stack and doing an RTS to effectively JMP to the function code; then each function code processor finished by doing a JMP $14E1 (my line 1520, label .1). My code effectively does a JSR to the function code, so each processor can finish by doing an RTS. This saves space, but is a tiny bit slower. However, I made up for the speed loss by eliminating one unnecessary range check. AppleWorks tested the function code range to be sure it was no larger than $11; if it was larger, AppleWorks jumped to HANG.UP. My code gives the same results, by merely extending the function code table in lines 4870-5190 to include all 32 vectors. The extra 28 bytes of table are more than saved elsewhere. By making the function code processors into subroutines which end with an RTS I have made them accessible to JSR calls from anywhere. This could save even more space.

You can see that to add more functions you merely have to write the processing subroutines and enter the vectors in the function code table using the >VEC macro. I can think of several neat additions. For example, I might use code 00 to initialize full screen, home, normal mode all in one code. It might also be useful to add a code to draw a file folder in the current window. A single code could shrink the window one notch, clear it, and draw a folder. Another code could pop back out to the next larger window. Let your imagination and creativity loose!

I wrote a little demonstration program, shown in lines 6040-6270. This program steps through a list of strings. Each string starts with a length byte. When a length byte of 00 is found, the demonstration stops. I have listed the demonstration strings in raw form to save paper, in lines 6290-6970. The demonstration is not too fancy, but it is fun. I display some characters, then move them around the screen in all four directions, with intermittent beeps to slow it down enough to see. Then I wipe it clean and display the entire character set.

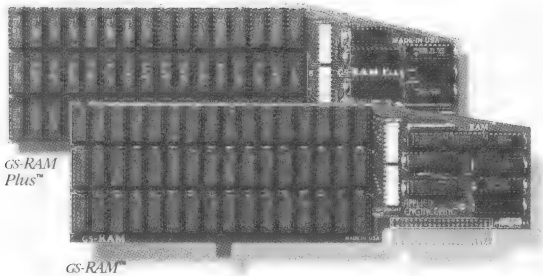
Let me know how you like this series of articles, and what kind of uses you find for the code. If you come up with some really great new function codes for DISPLAY.STRING, send them in and we'll share them in future issues.

```
                1010 *SAVE AW.SUBS.2
                1020 *-------------------------------
10-             1030 AW.LEFT     .EQ $10  DEFINES CURRENT WINDOW
11-             1040 AW.TOP      .EQ $11        "
12-             1050 AW.RIGHT    .EQ $12        "
13-             1060 AW.BOTTOM   .EQ $13        "
14-             1070 AW.CH       .EQ $14  CURSOR HORIZONTAL
15-             1080 AW.CV       .EQ $15  CURSOR VERTICAL
16-             1090 AW.BASE     .EQ $16,17
18-             1100 AW.BASE2    .EQ $18,19
                1110 *-------------------------------
F0-             1120 SHUFFLE.SOURCE       .EQ $F0
F1-             1130 SHUFFLE.DEST         .EQ $F1
                1140 *-------------------------------
                1150         .MA ST.SCRN
                1160         LSR          LSB into Carry
                1170         TAY          Index into Y-reg
                1180         PLA          Get char again
                1190         BCS :1       ...Odd character, in Main RAM
                1200         STA $C055    ...Even character, in Aux RAM
                1210 :1      STA (AW.BASE),Y
                1220         STA $C054
                1230         .EM
                1240 *-------------------------------
                1250         .MA LD.SCRN
                1260         LSR          LSB into Carry
                1270         TAY          Index into Y-reg
                1280         BCS :1       ...Odd character, in Main RAM
                1290         STA $C055    ...Even character, in Aux RAM
                1300 :1      LDA (AW.BASE),Y
                1310         STA $C054
                1320         .EM
                1330 *-------------------------------
0800-           1340 INVERSE.FLAG .BS 1  00=display chars inverse, 80=display
0801-           1350 BYTES.IN.STRING     .BS 1                      chars normal
0802-           1360 POSITION.IN.STRING  .BS 1
0803-           1370 SCROLL.DIRECTION    .BS 1
```

```
                    1380 *--------------------------------
                    1390 *   DISPLAY A STRING WITH FUNCTION CODES
                    1400 *      (A) = # BYTES IN STRING
                    1410 *      ($80,81) = Address of string
                    1420 *      ($A4) = flag:  if 00 display, else ignore.
                    1430 *   at $14D1 in AppleWorks 1.3
                    1440 *--------------------------------
                    1450 DISPLAY.STRING
0804- 8D 01 08      1460        STA BYTES.IN.STRING
0807- A5 A4         1470        LDA $A4         If non-zero, don't display anything
0809- D0 5B         1480        BNE .99         ...to an RTS
080B- 8D 02 08      1490        STA POSITION.IN.STRING
080E- 20 3A 0A      1500        JSR POLL.KEYBOARD    GET KEY IF ANY
                    1510 *--------------------------------
0811- AC 02 08      1520 .1     LDY POSITION.IN.STRING
0814- CC 01 08      1530        CPY BYTES.IN.STRING
0817- B0 4D         1540        BCS .99         ...to an RTS
0819- EE 02 08      1550        INC POSITION.IN.STRING
081C- B1 80         1560        LDA ($80),Y  GET CHAR FROM STRING
081E- 30 15         1570        BMI .2          ...80-FF, Mouse char for screen
0820- C9 20         1580        CMP #$20
0822- 90 32         1590        BCC .5          ...00-1F
0824- 4D 00 08      1600        EOR INVERSE.FLAG
0827- 30 10         1610        BMI .3          ...Normal Char
0829- C9 40         1620        CMP #$40        ...Inverse char
082B- 90 0C         1630        BCC .3          ...00-3F, inverse A-Z, 0-9, punctuation
082D- C9 60         1640        CMP #$60
082F- B0 08         1650        BCS .3          ...60-7F, inverse a-z, punctuation
0831- 29 BF         1660        AND #$BF        map 40-5F into 00-1F, more A-Z
0833- 90 04         1670        BCC .3          ...always
                    1680 *---Display a Mouse Char---------
0835- 29 7F         1690 .2     AND #$7F        ...map into 40-5F, mouse char range
0837- 09 40         1700        ORA #$40
                    1710 *---Display Char in A-register---
0839- 48            1720 .3     PHA             Save the character
083A- 20 3B 0A      1730        JSR BASE.CALC.CV
083D- A5 14         1740        LDA AW.CH       Char position on line
083F-               1750        >ST.SCRN        Store character on screen
                    1760 *--------------------------------
084C- A6 14         1770        LDX AW.CH
084E- E0 4F         1780        CPX #79         End of line yet?
0850- B0 BF         1790        BCS .1          ...yes, stick there, pile em' up
0852- E6 14         1800        INC AW.CH       ...no, advance CH
0854- D0 BB         1810        BNE .1          ...always
                    1820 *--------------------------------
0856- 20 5C 08      1830 .5     JSR .6          CALL THE FUNCTION
0859- 4C 11 08      1840        JMP .1          ...ALWAYS
                    1850 *--------------------------------
085C- 0A            1860 .6     ASL             Convert code to index
085D- AA            1870        TAX
085E- BD FB 09      1880        LDA FUNTBL+1,X
0861- 48            1890        PHA
0862- BD FA 09      1900        LDA FUNTBL,X
0865- 48            1910        PHA
0866- 60            1920 .99    RTS
                    1930 *--------------------------------
                    1940 *   FUNCTION 03 -- CLEAR ENTIRE WINDOW, CURSOR TO TOP-LEFT
                    1950 *   at $154A in AppleWorks 1.3
                    1960 *--------------------------------
                    1970 FUN.HOME
0867- A5 10         1980        LDA AW.LEFT
0869- 85 14         1990        STA AW.CH
086B- A5 11         2000        LDA AW.TOP
086D- 85 15         2010        STA AW.CV
                    2020 *--------------------------------
                    2030 *   FUNCTION 04 -- CLEAR REST OF WINDOW
                    2040 *   at $1552 in AppleWorks 1.3
                    2050 *--------------------------------
                    2060 FUN.CLR.CH.TO.EOS
086F- A5 14         2070        LDA AW.CH    SAVE CH AND CV
0871- 48            2080        PHA
0872- A5 15         2090        LDA AW.CV
0874- 48            2100        PHA
0875- 20 5D 0A      2110 .1     JSR CLR.CH.TO.EOL
0878- A5 15         2120        LDA AW.CV
087A- C5 13         2130        CMP AW.BOTTOM
087C- B0 08         2140        BCS .2          ...THAT WAS THE BOTTOM LINE
087E- E6 15         2150        INC AW.CV
0880- A5 10         2160        LDA AW.LEFT
0882- 85 14         2170        STA AW.CH
0884- 90 EF         2180        BCC .1          ...ALWAYS
0886- 68            2190 .2     PLA
```

For Those Who Want the Most. From Those Who Make the Best. GS-RAM.™

Now expand the IIGS' RAM and ROM with up to 8 MEG of "Instant On" memory with the all new GS-RAM!



GS-RAM has an all new design. A design that delivers higher performance including increased speed, greater expandability, and improved software.

More Sophisticated, Yet Easier to Use

GS-RAM works with all IIGS software. In fact *any* program that runs on Apple's smaller memory card runs on the GS-RAM. But with GS-RAM you can have more memory, improved performance, and almost unlimited expansion capabilities. We've designed the new GS-RAM to be easier to use too—you don't have to adjust the size of your RAM disk every time you use a DMA device. No other RAM card with more than 4 banks of memory installed can make the same claim.

More than Just Hardware

Each GS-RAM and GS-RAM Plus includes the most powerful set of IIGS software enhancements available anywhere. In fact, our nearest competitor offers only a fraction of the invaluable programs that we include with each GS-RAM card. This software includes the most powerful disk-caching program available, the GS-RAM Cache. The Cache will make most of your applications run up to 7 *times faster.* Also included is a diagnostic utility that lets you test your GS-RAM by graphically showing the location of any bad or improperly installed RAM chips. And for AppleWorks users, we give you our exclusive Expander program that dramatically enhances both the capabilities and speed of AppleWorks.

Making AppleWorks Even Better

Applied Engineering's Expander program eliminates AppleWorks internal memory limits allowing it to recognize up to 8 megabytes of desktop workspace. You can increase the limits from only 7,250 lines to 22,600 lines in the word processor and from 6,350 records to 22,600 records in the database. The Expander allows all of AppleWorks, including print functions, to automatically load into RAM. The clipboard size will increase from 255 to 2,042 lines maximum. GS-RAM will automatically segment larger files so you can save them onto multiple floppies. And GS-RAM provides a built-in print buffer that allows you to continue working in AppleWorks while your printer is still processing text. You can even load Pinpoint or MacroWorks and your favorite spelling checker into RAM for instant response.

Grow by Kilobytes or Megabytes

We offer GS-RAM in two configurations so you can increase your memory 256K at a time (GS-RAM) or a megabyte at a time (GS-RAM Plus). Both are IIGS compatible and both come with our powerful enhancement software. GS-RAM can hold up to 1.5 MEG of 256K chips and GS-RAM Plus can hold up to 6 MEG using 1 MEG chips. And since both use standard RAM chips (not high-priced SIMM's), you'll find expanding your GS-RAM or GS-RAM Plus easy, convenient, and very economical. For further expansion, you can plug a 2 MEG "piggyback" card into the GS-RAM's expansion port for up to 3.5 MEG of total capacity. Or up to a whopping 8 MEG on GS-RAM Plus. If a GS-RAM owner outgrows 3.5 MEG, he can easily upgrade to GS-RAM Plus for a nominal charge.

Permanent Storage for an "Instant On" Apple

With our RamKeeper™ back-up option, your GS-RAM or GS-RAM Plus will retain both programs and data while your IIGS is turned off! Now when you turn your IIGS back on, your favorite software is on your screen in under 4 seconds! With RamKeeper you can divide your IIGS memory into part "electronic hard disk," and part extended RAM. Even change the memory boundaries at any time—and in any way—you want. Because Applied Engineering has the most experience in the industry with battery-backed memory for the Apple, you are assured of the most reliable memory back-up system available. And in the world of battery-backed memory, *Reliability* is everything. That's why Applied Engineering uses state-of-the-art "GEL-CELL's" instead of Ni-Cad batteries (if Ni-Cads aren't discharged periodically, they lose much of their capacity). RamKeeper has about 6 hours of "total power failure" back-up time. That's 6 times the amount of other systems. But with power from your wall outlet, RamKeeper will back-up GS-RAM, GS-RAM Plus, or most other IIGS memory cards indefinitely. Should you ever have a "total power failure," RamKeeper switches to its 6-hour battery. When power returns, RamKeeper will automatically recharge the battery to full power. RamKeeper incorporates a dual-rate charger, status L.E.D.'s, and advanced power reducing circuitry. RamKeeper comes *complete* with battery, software, and documentation.

"In quality, performance, compatibility, expandability and support, Applied Engineering's GSRAM and GSRAM Plus are number one."

Steve Wozniak, the creator of Apple Computer

GS-RAM's Got it ALL!

- 5-year warranty — parts & labor
- 6 RAM banks (most cards have 4)
- Memory expansion port
- ROM expansion port
- Ultra-fast disk caching on ProDOS 8 *AND* ProDOS 16.
- Expands AppleWorks internal limits
- Includes hi-res self test
- No soldered-in RAM chips
- Expandable to 8 MEG
- No configuration blocks to set
- RamKeeper back-up option allows permanent storage of programs & data
- 15-day money-back guarantee
- Proudly made in the U.S.A.

GS-RAM with 256K	**$189**
GS-RAM with 512K	**$259**
GS-RAM with 1 MEG	**$399**
GS-RAM with 1.5 MEG	**$539**
GS-RAM with 2.5 to 3.5 MEG	**CALL**
GS-RAM Plus with 1-8 MEG	**CALL**
RamKeeper Option	**$179**

Order today!

See your dealer or call Applied Engineering today, 9 a.m. to 11 p.m. 7 days. Or send check or money order to Applied Engineering. MasterCard, VISA and C.O.D. welcome. Texas residents add 7% sales tax. Add $10.00 outside U.S.A.

APPLIED ENGINEERING™

The Apple enhancement experts.

(214) 241-6060

P.O. Box 798, Carrollton, TX 75006

Prices subject to change without notice.

GS-RAM, GS-RAM Plus and RamKeeper are trademarks of Applied Engineering. Other brands and product names are registered trademarks of their respective holders.

```
0887- 85 15     2200         STA AW.CV
0889- 68        2210         PLA
088A- 85 14     2220         STA AW.CH
088C- 60        2230         RTS
                2240 *-------------------------------
                2250 *   FUNCTION 05 -- GO TO X,Y IN THIS WINDOW
                2260 *      Equivalent to HTAB X, VTAB Y
                2270 *      05 XX YY in string
                2280 *   at $1576 in AppleWorks 1.3
                2290 *-------------------------------
                2300 FUN.GOTO.XY
088D- AC 02 08  2310         LDY POSITION.IN.STRING
0890- B1 80     2320         LDA ($80),Y
0892- C8        2330         INY
0893- CC 01 08  2340         CPY BYTES.IN.STRING
0896- B0 18     2350         BCS FUN.HANG.UP
0898- 65 10     2360         ADC AW.LEFT
089A- 85 14     2370         STA AW.CH
089C- B1 80     2380         LDA ($80),Y
089E- 65 11     2390         ADC AW.TOP
08A0- 85 15     2400         STA AW.CV
08A2- C8        2410         INY
08A3- 8C 02 08  2420         STY POSITION.IN.STRING
08A6- 60        2430         RTS
                2440 *-------------------------------
                2450 *   FUNCTION 06 -- BACK UP CURSOR
                2460 *   at $1593 in AppleWorks 1.3
                2470 *-------------------------------
                2480 FUN.CURSOR.LEFT
08A7- A5 14     2490         LDA AW.CH
08A9- C5 10     2500         CMP AW.LEFT
08AB- F0 03     2510         BEQ FUN.HANG.UP
08AD- C6 14     2520         DEC AW.CH
08AF- 60        2530         RTS
                2540 *-------------------------------
                2550 *   at $1599 and $1815 in AppleWorks 1.3
08B0- 4C B0 08  2560 FUN.HANG.UP JMP FUN.HANG.UP
                2570 *-------------------------------
                2580 *   FUNCTION 07 -- CURSOR RIGHT
                2590 *   at $15A1 in AppleWorks 1.3
                2600 *-------------------------------
                2610 FUN.CURSOR.RIGHT
08B3- A5 14     2620         LDA AW.CH
08B5- C5 12     2630         CMP AW.RIGHT
08B7- F0 F7     2640         BEQ FUN.HANG.UP
08B9- E6 14     2650         INC AW.CH
08BB- 60        2660         RTS
                2670 *-------------------------------
                2680 *   FUNCTION 08 -- CURSOR UP (SCROLL DOWN IF NECESSARY)
                2690 *   at $15AB in AppleWorks 1.3
                2700 *-------------------------------
                2710 FUN.CURSOR.UP
08BC- A5 15     2720         LDA AW.CV
08BE- C5 11     2730         CMP AW.TOP
08C0- F0 03     2740         BEQ .1       ...ALREADY AT TOP, SCROLL DOWN
08C2- C6 15     2750         DEC AW.CV
08C4- 60        2760         RTS
08C5- A5 13     2770 .1      LDA AW.BOTTOM
08C7- A2 FF     2780         LDX #-1
08C9- D0 0D     2790         BNE SCROLL   ...ALWAYS
                2800 *-------------------------------
                2810 *   FUNCTION 09 -- CURSOR DOWN (SCROLL UP IF NECESSARY)
                2820 *   at $15BC in AppleWorks 1.3
                2830 *-------------------------------
                2840 FUN.CURSOR.DOWN
08CB- A5 15     2850         LDA AW.CV
08CD- C5 13     2860         CMP AW.BOTTOM
08CF- F0 03     2870         BEQ .1       ...ALREADY AT BOTTOM, SCROLL UP
08D1- E6 15     2880         INC AW.CV
08D3- 60        2890         RTS
08D4- A5 11     2900 .1      LDA AW.TOP
08D6- A2 01     2910         LDX #1
                2920 SCROLL
08D8- 8E 03 08  2930         STX SCROLL.DIRECTION  01 IF UP, FF IF DOWN
08DB- 48        2940         PHA          SAVE LINE NUMBER
08DC- 20 3D 0A  2950         JSR BASE.CALC.A
08DF- A5 16     2960 .1      LDA AW.BASE
08E1- 85 18     2970         STA AW.BASE2
08E3- A5 17     2980         LDA AW.BASE+1
08E5- 85 19     2990         STA AW.BASE2+1
08E7- 68        3000         PLA          GET LINE NUMBER AGAIN
08E8- C5 15     3010         CMP AW.CV    IS IT THE LAST LINE?
```

```
08EA- F0 2A     3020         BEQ FUN.CLR.LINE
08EC- 18        3030         CLC
08ED- 6D 03 08  3040         ADC SCROLL.DIRECTION
08F0- 48        3050         PHA             SAVE SOURCE LINE NUMBER
08F1- 20 3D 0A  3060         JSR BASE.CALC.A
08F4- A5 10     3070         LDA AW.LEFT
08F6- AA        3080         TAX
08F7- 4A        3090         LSR
08F8- A8        3100         TAY
08F9- B0 0F     3110         BCS .3          START WITH ODD COLUMN
08FB- 8D 55 C0  3120 .2      STA $C055       EVEN COLUMNS IN AUX MEM
08FE- B1 16     3130         LDA (AW.BASE),Y
0900- 91 18     3140         STA (AW.BASE2),Y
0902- 8D 54 C0  3150         STA $C054       BACK TO MAIN MEM
0905- E4 12     3160         CPX AW.RIGHT
0907- F0 D6     3170         BEQ .1          ...END OF THIS LINE
0909- E8        3180         INX
090A- B1 16     3190 .3      LDA (AW.BASE),Y
090C- 91 18     3200         STA (AW.BASE2),Y
090E- E4 12     3210         CPX AW.RIGHT
0910- F0 CD     3220         BEQ .1          ...END OF THIS LINE
0912- E8        3230         INX
0913- C8        3240         INY
0914- D0 E5     3250         BNE .2          ...ALWAYS
                3260 *--------------------------------
                3270 *   FUNCTION 02 -- CLEAR ENTIRE CURRENT LINE
                3280 *   at $1540 in AppleWorks 1.3
                3290 *--------------------------------
                3300 FUN.CLR.LINE
0916- A5 10     3310         LDA AW.LEFT
0918- 85 14     3320         STA AW.CH
                3330 *--------------------------------
                3340 *   FUNCTION 01 -- CLEAR REST OF CURRENT LINE
                3350 *   at $1544 in AppleWorks 1.3
                3360 *--------------------------------
                3370 FUN.CLR.CH.TO.EOL
091A- 4C 5D 0A  3380         JMP CLR.CH.TO.EOL
                3390 *--------------------------------
                3400 *   FUNCTION 0A -- INVERSE
                3410 *   at $160B in AppleWorks 1.3
                3420 *--------------------------------
                3430 FUN.INVERSE
091D- A9 00     3440         LDA #$00
091F- 2C        3450         .HS 2C          SKIP OVER LDA #$80
                3460 *--------------------------------
                3470 *   FUNCTION 0B -- NORMAL
                3480 *   at $160F in AppleWorks 1.3
                3490 *--------------------------------
                3500 FUN.NORMAL
0920- A9 80     3510         LDA #$80
0922- 8D 00 08  3520         STA INVERSE.FLAG
0925- 60        3530         RTS
                3540 *--------------------------------
                3550 *   FUNCTION 0C -- DEFINE BOTTOM RIGHT CORNER
                3560 *   at $1617 in AppleWorks 1.3
                3570 *--------------------------------
                3580 FUN.CORNER.BR
0926- A5 14     3590         LDA AW.CH
0928- A6 15     3600         LDX AW.CV
                3610 SET.CORNER.BR
092A- 85 12     3620         STA AW.RIGHT
092C- 86 13     3630         STX AW.BOTTOM
092E- 60        3640         RTS
                3650 *--------------------------------
                3660 *   FUNCTION 0D -- CURSOR TO BEGINNING OF LINE
                3670 *      Equivalent to RETURN without LINEFEED
                3680 *   at $1622 in AppleWorks 1.3
                3690 *--------------------------------
                3700 FUN.CURSOR.BOL
092F- A5 10     3710         LDA AW.LEFT
0931- 85 14     3720         STA AW.CH
0933- 60        3730         RTS
                3740 *--------------------------------
                3750 *   FUNCTION 0E -- DEFINE TOP LEFT CORNER
                3760 *   at $1629 in AppleWorks 1.3
                3770 *--------------------------------
                3780 FUN.CORNER.TL
0934- A5 14     3790         LDA AW.CH
0936- A6 15     3800         LDX AW.CV
0938- 85 10     3810         STA AW.LEFT
093A- 86 11     3820         STX AW.TOP
093C- 60        3830         RTS
```

```
                    3840 *-------------------------------
                    3850 *   FUNCTION 0F -- SET FULL SCREEN WINDOW
                    3860 *   at $1634 in AppleWorks 1.3
                    3870 *-------------------------------
                    3880 FUN.FULL.SCREEN
093D- A9 00         3890         LDA #0
093F- 85 11         3900         STA AW.TOP
0941- 85 10         3910         STA AW.LEFT
0943- A9 4F         3920         LDA #79
0945- A2 17         3930         LDX #23
0947- D0 E1         3940         BNE SET.CORNER.BR   ...ALWAYS
                    3950 *-------------------------------
                    3960 *   FUNCTION 10 -- "BEEP!"
                    3970 *   at $1645 in AppleWorks 1.3
                    3980 *-------------------------------
                    3990 FUN.BEEP
0949- A0 95         4000         LDY #149
094B- A9 95         4010 .1      LDA #149
094D- 38            4020 .2      SEC
094E- E9 01         4030         SBC #1
0950- 38            4040         SEC
0951- 38            4050         SEC
0952- D0 F9         4060         BNE .2         11*149-1 = 1638 CYCLES IN INNER LOOP
0954- AD 30 C0      4070         LDA $C030      TOGGLE SPEAKER
0957- 88            4080         DEY
0958- D0 F1         4090         BNE .1         1649 CYCLES BETWEEN CLICKS
095A- A9 01         4100         LDA #1         DELAY ABOUT 1/10 SECOND
095C- 4C 9B 0A      4110         JMP DELAY.TENTHS
                    4120 *-------------------------------
                    4130 *   FUNCTION 11 -- SHUFFLE SCREEN LEFT OR RIGHT
                    4140 *      Following byte is scroll distance and direction:
                    4150 *      If positive, SCROLL RIGHT; else SCROLL LEFT.
                    4160 *   at $165E in AppleWorks 1.3
                    4170 *-------------------------------
                    4180 FUN.SHUFFLE
095F- AC 02 08      4190         LDY POSITION.IN.STRING
0962- B1 80         4200         LDA ($80),Y  If positive, SCROLL RIGHT; else SCROLL LEFT
0964- 8D 03 08      4210         STA SCROLL.DIRECTION
0967- EE 02 08      4220         INC POSITION.IN.STRING
096A- A5 11         4230         LDA AW.TOP   POINT TO TOP LINE FIRST
096C- 48            4240 .1      PHA
096D- 20 3D 0A      4250         JSR BASE.CALC.A
0970- A5 12         4260         LDA AW.RIGHT
0972- 2C 03 08      4270         BIT SCROLL.DIRECTION
0975- 10 02         4280         BPL .2
0977- A5 10         4290         LDA AW.LEFT
0979- 85 F1         4300 .2      STA SHUFFLE.DEST
097B- 38            4310         SEC
097C- ED 03 08      4320         SBC SCROLL.DIRECTION
097F- 85 F0         4330         STA SHUFFLE.SOURCE
0981- 20 8F 09      4340         JSR SCROLL.LEFT.OR.RIGHT
0984- 68            4350         PLA
0985- C5 13         4360         CMP AW.BOTTOM
0987- F0 05         4370         BEQ .3
0989- 18            4380         CLC
098A- 69 01         4390         ADC #1
098C- D0 DE         4400         BNE .1
098E- 60            4410 .3      RTS
                    4420 *-------------------------------
                    4430 SCROLL.LEFT.OR.RIGHT
098F- AD 03 08      4440         LDA SCROLL.DIRECTION
0992- 10 38         4450         BPL .6         ...shuffle right
0994- 30 04         4460         BMI .2         ...shuffle left
                    4470 *---Scroll Left----------------
0996- E6 F1         4480 .1      INC SHUFFLE.DEST
0998- E6 F0         4490         INC SHUFFLE.SOURCE
099A- A9 A0         4500 .2      LDA #" "       blank, in case off edge
099C- A4 F0         4510         LDY SHUFFLE.SOURCE
099E- C4 12         4520         CPY AW.RIGHT
09A0- 90 02         4530         BCC .3
09A2- D0 0D         4540         BNE .4         ...off the edge, use a blank
09A4- 98            4550 .3      TYA            Get source pointer
09A5-               4560         >LD.SCRN       Get character from screen
09B1- 48            4570 .4      PHA            Save the character
09B2- A5 F1         4580         LDA SHUFFLE.DEST  destination pointer
09B4-               4590         >ST.SCRN       Store the character on the screen
09C1- A5 F1         4600         LDA SHUFFLE.DEST  destination pointer
09C3- C5 12         4610         CMP AW.RIGHT      was it the right edge?
09C5- D0 CF         4620         BNE  .1           ...no, keep shuffling
09C7- 60            4630         RTS
```

```
                        4640 *---Scroll Right----------------
09C8- C6 F1             4650 .5     DEC SHUFFLE.DEST
09CA- C6 F0             4660        DEC SHUFFLE.SOURCE
09CC- A9 A0             4670 .6     LDA #" "      blank, in case off edge
09CE- A4 F0             4680        LDY SHUFFLE.SOURCE
09D0- 30 11             4690        BMI .7        ...off edge, use blank
09D2- C4 10             4700        CPY AW.LEFT
09D4- 90 0D             4710        BCC .7        ...off edge, use blank
09D6- 98                4720        TYA           Get source pointer
09D7-                   4730        >LD.SCRN      Get character from screen
09E3- 48                4740 .7     PHA           Save the character
09E4- A5 F1             4750        LDA SHUFFLE.DEST  destination pointer
09E6-                   4760        >ST.SCRN      Store the character on the screen
09F3- A5 F1             4770        LDA SHUFFLE.DEST  destination pointer
09F5- C5 10             4780        CMP AW.LEFT       was it the left edge?
09F7- D0 CF             4790        BNE  .5           ...no, keep shuffling
09F9- 60                4800        RTS
                        4810 *-------------------------------
                        4820        .MA VEC
                        4830        .DA FUN.]1-1
                        4840        .EM
                        4850 *-------------------------------
                        4860 *   at $1779 in AppleWorks 1.3
                        4870 FUNTBL
09FA-                   4880        >VEC HANG.UP          00
09FC-                   4890        >VEC CLR.CH.TO.EOL    01
09FE-                   4900        >VEC CLR.LINE         02
0A00-                   4910        >VEC HOME             03
0A02-                   4920        >VEC CLR.CH.TO.EOS    04
0A04-                   4930        >VEC GOTO.XY          05
0A06-                   4940        >VEC CURSOR.LEFT      06
0A08-                   4950        >VEC CURSOR.RIGHT     07
0A0A-                   4960        >VEC CURSOR.UP        08
0A0C-                   4970        >VEC CURSOR.DOWN      09
0A0E-                   4980        >VEC INVERSE          0A
0A10-                   4990        >VEC NORMAL           0B
0A12-                   5000        >VEC CORNER.BR        0C
0A14-                   5010        >VEC CURSOR.BOL       0D
0A16-                   5020        >VEC CORNER.TL        0E
0A18-                   5030        >VEC FULL.SCREEN      0F
0A1A-                   5040        >VEC BEEP             10
0A1C-                   5050        >VEC SHUFFLE          11
0A1E-                   5060        >VEC HANG.UP          12
0A20-                   5070        >VEC HANG.UP          13
0A22-                   5080        >VEC HANG.UP          14
0A24-                   5090        >VEC HANG.UP          15
0A26-                   5100        >VEC HANG.UP          16
0A28-                   5110        >VEC HANG.UP          17
0A2A-                   5120        >VEC HANG.UP          18
0A2C-                   5130        >VEC HANG.UP          19
0A2E-                   5140        >VEC HANG.UP          1A
0A30-                   5150        >VEC HANG.UP          1B
0A32-                   5160        >VEC HANG.UP          1C
0A34-                   5170        >VEC HANG.UP          1D
0A36-                   5180        >VEC HANG.UP          1E
0A38-                   5190        >VEC HANG.UP          1F
                        5200 *-------------------------------
0A3A- 60                5210 POLL.KEYBOARD RTS    (at $1FA7 in AppleWorks 1.3)
                        5220 *-------------------------------
                        5230 *   Calculate Address of Screen Line
                        5240 *   at $1717 in AppleWorks 1.3
                        5250 *-------------------------------
                        5260 BASE.CALC.CV
0A3B- A5 15             5270        LDA AW.CV
                        5280 BASE.CALC.A
0A3D- C9 FF             5290        CMP #$FF      <<<filled in with current line number>>>
0A3E-                   5300 CURR.LINE  .EQ *-1
0A3F- F0 1B             5310        BEQ .2
0A41- 8D 3E 0A          5320        STA CURR.LINE
0A44- 4A                5330        LSR           0000ABCD--E
0A45- 29 03             5340        AND #$03      000000CD--E
0A47- 09 04             5350        ORA #$04      000001CD--E
0A49- 85 17             5360        STA AW.BASE+1
0A4B- AD 3E 0A          5370        LDA CURR.LINE
0A4E- 29 18             5380        AND #$18      000AB000--E
0A50- 90 02             5390        BCC .1        E=0
0A52- 69 7F             5400        ADC #$7F      E00AB000
0A54- 85 16             5410 .1     STA AW.BASE
0A56- 0A                5420        ASL           00AB0000
0A57- 0A                5430        ASL           0AB00000
0A58- 05 16             5440        ORA AW.BASE   EABAB000
0A5A- 85 16             5450        STA AW.BASE
0A5C- 60                5460 .2     RTS
```

Now Apple speaks IBM. Three times faster than IBM.

Introducing PC Transporter.™ The Apple® II expansion board that lets you run MS®-DOS programs.

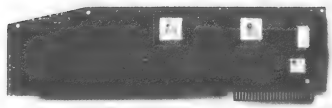
Now your Apple II can run over 10,000 programs you could never use before. Like Lotus® 1-2-3® MultiMate® dBASE III PLUS® Even Flight Simulator®

With PC Transporter, MS-DOS programs run on your Apple II like they do on IBM® PC's or compatibles. With one important difference. PC Transporter runs most of those programs *three times faster* than an IBM PC/XT®

Plus, to speed through number-crunching tasks, you can use our optional 8087-2 math co-processor chip. It plugs into a socket on the PC Transporter.

Less expensive than an IBM clone.

Sure, a stripped-down IBM



clone costs about the same as the PC Transporter. But the peripherals it takes to get the clone up and running make the clone cost about three times what our American-made card costs.

You don't have to buy new hardware to use PC Transporter.

Works with the hardware you already own.

With PC Transporter, MS-DOS programs see your Apple hardware as IBM hardware. You can use the same hardware you have now.

With IBM software, your Apple hardware works just like IBM hardware. Including your drives, monitors, printers, printer cards, clock cards and serial clocks.

You can use your IIe® or IIGS™ keyboard with IBM software. Or use our optional IBM-style keyboard (required for the II Plus).

You can use your Apple mouse. Or an IBM compatible serial mouse.

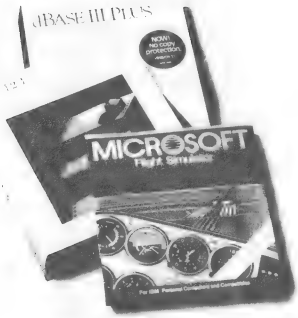
Plenty of power.

PC Transporter gives you as much as 640K of user RAM and 128K of system RAM in the IBM mode.

PC Transporter also is an Apple expansion card, adding up to 768K of extra RAM in the Apple mode. The Apple expansion alone is a $300 value.

Easy to install.

You can install PC Transporter in about 15 minutes, even if you've never added an expansion board. You don't need special tools. Simply plug it into an Apple expansion slot (1 through 7 except 3), connect a few cables and a disk drive, and go!



PC Transporter taps into the world's largest software library. Now your Apple can run most of the IBM software you use at work. And it opens a new world of communications programs, games and bulletin boards.

A universal disk drive controller.

PC Transporter supports 3.5" and 5.25" MS-DOS and ProDOS formatted diskettes. You'll shift instantly between Apple ProDOS and IBM MS-DOS.

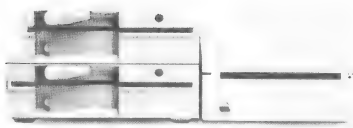
You'll need our versatile 5.25" 360K drive system to run IBM applications from 5.25" floppy disks. Use your Apple 5.25" drive for Apple 5.25" disks.

An Apple Disk 3.5 Drive will support the new 3.5"disks whether they're IBM MS-DOS formatted or Apple ProDOS formatted. The PC Transporter acts like an Apple Disk 3.5 Drive disk controller for IIGS, IIe, and II Plus users.

PC Transporter supports up to 5 drives in a number of combinations.

For example, you can connect a 5.25 Applied Engineering 360K dual-drive system directly to the card. Then plug two daisy-chained Apple 3.5 Drives (not the Apple UniDisk 3.5) to the dual-drive system. For a fifth drive, use a ProDOS file as an IBM hard disk.

PC Transporter controls Apple and IBM compatible disk drives. It supports 3.5" and 5.25" MS-DOS and ProDOS formatted diskettes.



Versatile data storage.

PC Transporter reads MS-DOS and translates it into Apple native ProDOS. You can store IBM programs and data on any ProDOS storage device including the Apple 3.5 Drive, Apple UniDisk™ 3.5, Apple 5.25" drive, SCSI or ProDOS compatible hard drives. (You can use the Apple UniDisk 3.5 with its own controller card for storing programs and data,but not for directly booting an IBM formatted disk.)

You can even use our 360K PC compatible drive for ProDOS

Make your Apple speak IBM.

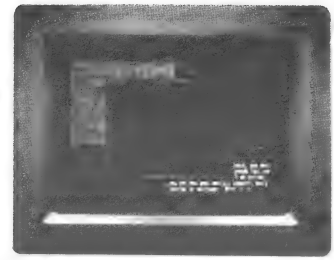
PC Transporter memory choices.

RAM in Apple mode:	RAM in IBM mode:	Price:
384K	**256K**	**$489.00**
512K	**384K**	**529.00**
640K	**512K**	**569.00**
768K	**640K**	**609.00**

Note: The IBM mode is 128K less because the PC Transporter uses 128K for system memory.

IIGS Installation Kit	**49.00**
IIe/II Plus Installation Kit	**39.00**
PC Transporter Accessories	
5.25" IBM Format 360K Drive Systems	
Single-Drive System	**269.00**
Dual-Drive System	**399.00**
Half-Height Drive (add to Single-Drive system to make Dual-Drive)	**135.00**
IBM-Style Keyboard (required for Apple II Plus. Requires IBM Keyboard Cable.)	**139.00**
IBM Keyboard Cable	**34.00**
Sony RGB Monitor	**499.00**
Analog RGB Cable (for use with Sony monitor)	**39.00**
Digital RGB Cable (for use with Sony monitor)	**39.00**
Digital RGB Adapter	**24.00**
ColorSwitch (included with IIGS Installation Kit)	**44.00**
128K ZIP PC Transporter Memory Expansion Chip Set	**40.00 per set**
8087-2 Math Co-processor Chip	**229.00**
Heavy Duty Power Supply (IIe and II Plus only)	**69.00**

See your dealer or call or send check or money order to Applied Engineering. MasterCard, VISA and COD welcome. Texas residents add 6¼% sales tax.



PC Transporter produces better IBM graphics than IBM. Analog is sharper than digital. So with an analog RGB monitor, PC Transporter's CGA graphics and text are superior to IBM's digital display — even while running IBM software!

And, you can also use an Apple composite monitor in IBM text or graphics mode.

storage and a 143K Apple 5.25" drive for MS-DOS storage.

Created by Apple's original designers.

The brains behind PC Transporter were also behind your Apple II.

The PC Transporter design team includes the former project managers for the creation of the Apple IIe and IIc. The co-designer of the Apple II disk controller. And the first full-time Apple programmer and author of the ProDOS operating system.

So you know the PC Transporter and your Apple were made for each other.

Support and service from the leader in Apple add-ons.

Applied Engineering sells more Apple peripheral boards than anyone else — including Apple Computer. So you know we'll be around after the sale.

PC Transporter comes with a 15-day money back guarantee. If you're not fully satisfied after using it, return it for a full refund. PC Transporter also comes with a 1-year warranty.

How to get your PC Transporter today.

See your dealer. Or call Applied Engineering any day between 9 a.m. and 11 p.m. CST at 214-241-6060.

Applied Engineering

The Apple enhancement experts.

P.O. Box 798, Carrollton, TX 75006
214-241-6060
A Division of AE Research Corporation

```
                        5470 *--------------------------------
                        5480 *   Blank Line from Cursor to End of Line
                        5490 *   at $173A in AppleWorks 1.3
                        5500 *--------------------------------
                        5510 CLR.CH.TO.EOL
0A5D- A5 12             5520         LDA AW.RIGHT
0A5F- 4A                5530         LSR
0A60- 8D 99 0A          5540         STA N.OVER.2
0A63- 2A                5550         ROL             GET AW.RIGHT AGAIN
0A64- 38                5560         SEC
0A65- E9 01             5570         SBC #1
0A67- 4A                5580         LSR
0A68- 8D 9A 0A          5590         STA N.MINUS.1.OVER.2
0A6B- 20 3B 0A          5600         JSR BASE.CALC.CV
0A6E- A5 14             5610         LDA AW.CH
0A70- 4A                5620         LSR
0A71- 48                5630         PHA
                        5640 *---Clear Even Columns-----------
0A72- A8                5650         TAY
0A73- A9 A0             5660         LDA #" "
0A75- 8D 55 C0          5670         STA $C055       INTO AUX MEM
0A78- 90 01             5680         BCC .2
0A7A- C8                5690 .1      INY
0A7B- CC 99 0A          5700 .2      CPY N.OVER.2
0A7E- F0 02             5710         BEQ .3
0A80- B0 05             5720         BCS .4
0A82- 91 16             5730 .3      STA (AW.BASE),Y
0A84- 4C 7A 0A          5740         JMP .1
0A87- 8D 54 C0          5750 .4      STA $C054       BACK TO MAIN MEM
                        5760 *---Clear Odd Columns------------
0A8A- 68                5770         PLA
0A8B- A8                5780         TAY
0A8C- A9 A0             5790         LDA #" "
0A8E- 91 16             5800 .5      STA (AW.BASE),Y
0A90- C8                5810         INY
0A91- CC 9A 0A          5820         CPY N.MINUS.1.OVER.2
0A94- 90 F8             5830         BCC .5
0A96- F0 F6             5840         BEQ .5
0A98- 60                5850         RTS
                        5860 *--------------------------------
0A99-                   5870 N.OVER.2            .BS 1
0A9A-                   5880 N.MINUS.1.OVER.2    .BS 1
                        5890 *--------------------------------
                        5900 *   DELAY ABOUT (A) TENTHS OF A SECOND
                        5910 *   at $1FD1 in AppleWorks 1.3
                        5920 *--------------------------------
                        5930 DELAY.TENTHS
0A9B- A8                5940         TAY
0A9C- A2 64             5950 .1      LDX #100        ...ABOUT 100 MILLISECONDS
0A9E- 18                5960 .2      CLC
0A9F- 69 01             5970 .3      ADC #1
0AA1- 90 FC             5980         BCC .3          ...LOOP IS 256*5 CYCLES
0AA3- CA                5990         DEX
0AA4- D0 F8             6000         BNE .2          (5*256+6)*100 = 128600 CYCLES
0AA6- 88                6010         DEY
0AA7- D0 F3             6020         BNE .1          128606*A CYCLES
0AA9- 60                6030         RTS
                        6040 *--------------------------------
                        6050 *   Some Demonstrations
                        6060 *--------------------------------
                        6070 T
0AAA- A9 CE             6080         LDA #MY.STRINGS
0AAC- A2 0A             6090         LDX /MY.STRINGS
0AAE- 85 80             6100 .1      STA $80
0AB0- 86 81             6110         STX $81
0AB2- A0 00             6120         LDY #0
0AB4- B1 80             6130         LDA ($80),Y
0AB6- F0 15             6140         BEQ .99         ...FINISHED
0AB8- 48                6150         PHA             SAVE LENGTH
0AB9- E6 80             6160         INC $80
0ABB- D0 02             6170         BNE .2
0ABD- E6 81             6180         INC $81
0ABF- 20 04 08          6190 .2      JSR DISPLAY.STRING
0AC2- 18                6200         CLC
0AC3- 68                6210         PLA             GET LENGTH
0AC4- 65 80             6220         ADC $80         ADD TO POINTER
0AC6- A6 81             6230         LDX $81
0AC8- 90 E4             6240         BCC .1
0ACA- E8                6250         INX
0ACB- B0 E1             6260         BCS .1          ...ALWAYS
0ACD- 60                6270 .99     RTS
```

```
6280 *--------------------------------
6290 MY.STRINGS
6300         .DA #Z1
6310 A1      .HS 0F.03.0B      Full Scrn, Home, Normal
6320         .AS /ABCDECDEFGHIJKLMNOPQRSTUVWXYZ !@#$%^&*()-+=[]\;:'",.<>?{}|/
6330         .HS 0A.0D.09      Inverse, RETURN, LINEFEED
6340         .AS /ABCDECDEFGHIJKLMNOPQRSTUVWXYZ !@#$%^&*()-+=[]\;:'",.<>?{}|/
6350 Z1      .EQ *-A1
6360 *--------------------------------
6370         .DA #Z2
6380 A2      .HS 0B.0D.09      Normal, RETURN, LINEFEED
6390         .AS "abcdefghijklmnopqrstuvwxyz / 0123456789 _ "
6400         .HS 0A.0D.09      Inverse, RETURN, LINEFEED
6410         .AS "abcdefghijklmnopqrstuvwxyz / 0123456789 _ "
6420 Z2      .EQ *-A2
6430 *--------------------------------
6440         .DA #Z3
6450 A3      .HS 0B.0D.09      Normal, RETURN, LINEFEED
6460         .AS -/@ABCDECDEFGHIJKLMNOPQRSTUVWXYZ[\]^_/
6470         .HS 05.00.00      GO TO 0,0
6480         .HS 10.08.08.08.08.08.08.08  Beep, 7 cursor ups
6490         .HS 10.08.08.08.08.08.08.08  Beep, 7 cursor ups
6500         .HS 05.00.17      GO TO 0,23
6510         .HS 10.09.09.09.09.09.09.09  Beep, 7 cursor downs
6520         .HS 10.09.09.09.09.09.09.09  Beep, 7 cursor downs
6530 Z3      .EQ *-A3
6540 *--------------------------------
6550         .DA #Z4
6560 A4      .HS 10.11.08      Beep, Shuffle Right 8
6570         .HS 10.11.F8      Beep, Shuffle Left 8
6580         .HS 10.11.01.11.01.11.01.11.01.11.01.11.01.11.01.11
6590         .HS 10.11.FF.11.FF.11.FF.11.FF.11.FF.11.FF.11.FF.11.FF
6600         .HS 10.11.F8
6610         .HS 10.11.08
6620         .HS 10.11.28
6630         .HS 10.11.D8
6640 Z4      .EQ *-A4
6650 *--------------------------------
6660         .DA #Z5
6670 A5      .HS 03            HOME
6680         .HS 20.21.22.23.24.25.26.27.28.29.2A.2B.2C.2D.2E.2F
6690         .HS 30.31.32.33.34.35.36.37.38.39.3A.3B.3C.3D.3E.3F.0D.09
6700         .HS 40.41.42.43.44.45.46.47.48.49.4A.4B.4C.4D.4E.4F
6710         .HS 50.51.52.53.54.55.56.57.58.59.5A.5B.5C.5D.5E.5F.0D.09
6720         .HS 60.61.62.63.64.65.66.67.68.69.6A.6B.6C.6D.6E.6F
6730         .HS 70.71.72.73.74.75.76.77.78.79.7A.7B.7C.7D.7E.7F.0D.09
6740         .HS 0A            INVERSE
6750         .HS 20.21.22.23.24.25.26.27.28.29.2A.2B.2C.2D.2E.2F
6760         .HS 30.31.32.33.34.35.36.37.38.39.3A.3B.3C.3D.3E.3F.0D.09
6770         .HS 40.41.42.43.44.45.46.47.48.49.4A.4B.4C.4D.4E.4F
6780         .HS 50.51.52.53.54.55.56.57.58.59.5A.5B.5C.5D.5E.5F.0D.09
6790         .HS 60.61.62.63.64.65.66.67.68.69.6A.6B.6C.6D.6E.6F
6800         .HS 70.71.72.73.74.75.76.77.78.79.7A.7B.7C.7D.7E.7F.0D.09
6810         .HS 0B            NORMAL
6820 Z5      .EQ *-A5
6830 *--------------------------------
6840         .DA #Z6
6850 A6      .HS 0D.09         CRLF
6860         .HS 80.81.82.83.84.85.86.87.88.89.8A.8B.8C.8D.8E.8F
6870         .HS 90.91.92.93.94.95.96.97.98.99.9A.9B.9C.9D.9E.9F.0D.09
6880         .HS A0.A1.A2.A3.A4.A5.A6.A7.A8.A9.AA.AB.AC.AD.AE.AF
6890         .HS B0.B1.B2.B3.B4.B5.B6.B7.B8.B9.BA.BB.BC.BD.BE.BF.0D.09
6900         .HS C0.C1.C2.C3.C4.C5.C6.C7.C8.C9.CA.CB.CC.CD.CE.CF
6910         .HS D0.D1.D2.D3.D4.D5.D6.D7.D8.D9.DA.DB.DC.DD.DE.DF.0D.09
6920         .HS E0.E1.E2.E3.E4.E5.E6.E7.E8.E9.EA.EB.EC.ED.EE.EF
6930         .HS F0.F1.F2.F3.F4.F5.F6.F7.F8.F9.FA.FB.FC.FD.FE.FF.0D.09
6940 Z6      .EQ *-A6
6950 *--------------------------------
6960         .HS 00
6970 *--------------------------------
```

Overhauling the S-C Program Selector.....Bob Sander-Cederlof

About a year and a half ago, in the July 1986 issue of Apple Assembly Line, I published my replacement for the ProDOS "QUIT" code. It turned out to be a very popular article and program, and various updates and corrections were printed in the August, September, October, and December issues that same year.

In review, the QUIT code is a 768-byte program inside ProDOS-8 which is executed when you leave a system program, such as FILER, AppleWorks, BASIC.SYSTEM (Applesoft), Copy II Plus, the S-C Macro Assembler, and so on. When you QUIT you are really executing the MLI Quit call ($65), which copies those 768 bytes down to RAM starting at $1000 and jumps there. Apple's built-in QUIT code is about as user-friendly as an angry porcupine: if you haven't MEMORIZED the volume names and file names of all your system programs, you almost always end up resetting and re-booting instead of filling in the blanks.

My 1986 Program Selector exactly fits in the 768-byte space Apple's version occupies in the ProDOS system file. It puts a menu of online volumes on the screen, allowing you to select one with the arrow keys and RETURN. Once you select a volume, you see a menu of the SYS and DIR files in the main directory of that volume. Using the arrows and RETURN you can either start up a SYS program or select a subdirectory. If you choose a subdirectory, you get a menu of SYS and DIR files inside it. Hitting ESCAPE always takes you back to the Volume Menu. As I wrote the Program Selector, it requires an Apple //e, //c, or //gs, and works in 80-columns.

The modifications already published include fixing one bug, making it work with a Videx-compatible 80-column card for Apple II Plus owners, following Apple's published spec's for substitute QUIT-code, and making it begin with the menu bar on something other than the first volume in the menu.

Quite a few readers of AAL are now using this Program Selector. Some have gone the extra mile by adapting the S-C Program Selector to their own preferences. Jim Hammond (of FastFind SUPER INDEX) liked it so well he turned it into a product which he sells (with my permission) as "STARTER/QUITTER". Larry Skutchan (a blind user who adapted the S-C Word Processor into a talking version) adapted it to work with the Echo Speech Synthesizer. Brooke Boering (creator of CeeMac and Fire Organ) put in a feature allowing you to limit the Volume Menu to a particular disk drive. Brooke's ideas are what led me to try to improve and upgrade my program.

The main computer here at the office is an Apple //e with a 10-meg Sider (ProDOS sees it as two drives in slot 7), two standard Apple floppies in slot 6, a RamFactor card in slot 4 (simulates one drive), a 1-meg RamWorks card in the AuxSlot (simulates a drive in slot 3, Drive 2), and a 3.5 inch drive in slot 2 (ProDOS thinks there are two drives there, even though I only have one). When I type BYE or in some other way select the ProDOS QUIT code, my S-C Program Selector takes over. The old version seemed to go away and die for several seconds while

ProDOS did a complete ONLINE check to find out what volume if any was mounted in each and every drive. If my hard disk is not turned on, that takes several seconds for the firmware to timeout. If no floppies are in the 5.25 drives, they go through spinning, re-calibration, and the works before giving up.

It finally dawned on me that what I wanted was to direct the Program Selector to only try the slot and drive I booted from. Most likely if I booted from it, there is some kind of volume there. Of course I still wanted the capability of seeing every volume, but I did not want to waste all that time EVERY time!

Brooke told me six months or more ago that I could plug a drive ID into my ONLINE call (line 2760 of the original program) and it would limit the display to that one volume. It turned out to be a little harder than that, but I did get that to work. But I could not decide on just one slot and drive. I wanted that byte to be set up by ProDOS at boot-time to point to the booting drive.

I remembered that the ProDOS startup code began by plugging the boot drive ID into its own ONLINE call, which it uses to get the Volume Name. I looked up the code in the Supplement to "Beneath Apple ProDOS", and found it. In ProDOS 1.1.1 it is done by the first two instructions at $2000; in ProDOS 1.2, 1.3, and 1.4 it starts six bytes later at $2006. The first instruction is "LDA $43", which picks up the drive ID (slot number times 16) that was used to load the PRODOS or P8 file. The second one is "STA $21FE" for ProDOS 1.1.1, and "STA" somewhere else for later versions. I decided I could patch into that "STA" instruction a call to a piece of patch code which would not only do the patched-over STA for Apple, but also an additional one for me. More on this later, when we get into the code for my new Program Selector.

Anyway, the Program Selector now comes up only showing the Volume Name of the volume currently in the drive ProDOS was booted from. If that is the volume I want, I just hit RETURN and see the menu of SYS and DIR files on that volume. If I booted from RamFactor, this all happens at blinding speed.

If the boot-drive is not the drive I want, I can type a digit and select a different drive or all drives. Typing a digit 1 through 7 changes to drive 1 of that slot and displays the Volume Name in that drive. Typing the digit "8" changes to drive 2 in the same slot. Why 8? Why not? It made the code shorter, and it worked. Typing "0" changes back to the old way, making a menu of all online volumes. If you select a drive which has no volume mounted, you will get an empty menu. No problem, just select a different drive or all-drives, and continue.

I also made various other improvements here and there, such as making sure that text mode with a full-screen window is selected. I had to revise the "help" message at the bottom of the menu display to include information on the digits 0-8.

A major constraint in adding new features was that I wanted to retain the advantage of fitting inside the 768-byte hole in the ProDOS file. In developing the new version I decided not to worry about size too much until all the new features were working. I tested them by BRUNning the code at $1000, instead of going through the process of putting it into ProDOS every time. Then when it was all ready, I started looking for ways to shrink the code and make it fit in only 768 bytes.

It ran about 32 bytes over, so I needed a lot of shrinking. For some reason I don't remember, back in 1986 I decided to keep a lot of variables out of page zero. There is no requirement to do this, so I moved these variables and saved almost all the bytes I wanted. All instructions referencing these variables shrank from three to two bytes. The rest of the savings were found by careful study of the code. If you compare the new listing which follows with the one I published in 1986 you can find the tricks I pulled. The new version, even with all the new features, is now shorter than the original! There are actually four unused bytes!

I also wrote a program to automatically install the new Program Selector inside the ProDOS file. Well, almost automatically. You still have to BLOAD PRODOS or BLOAD P8, and UNLOCK the file if it is LOCKed. Then you "-" or BRUN my INSTALL.QUITTER program, and it automatically does the installation. If successful, you get a nice message to that effect; then you have to BSAVE the image and re-LOCK it. I thought it would be too dangerous to make all of the above entirely automatic. If my installer made a mistake.... So, I left the crucial part manual.

The auto-installer does differentiate between ProDOS 1.1.1 and the later versions. It makes the boot-drive patch at either $2002 or $2008, depending on where it finds the STA instruction. And if it cannot find that instruction, it tells you so and quits. The Program Selector image is copied either to $5700 (for ProDOS 1.1.1) or $5900 (for later versions).

The code for the auto-installer is executed when you BRUN INSTALL.QUITTER. Lines 1460-2270 are the installation code, and lines 2280 to the end are the Program Selector image. Lines 1490-1710 try to determine which version of ProDOS, if any, is in memory starting at $2000. If it finds a recognizable version, it sets up various pointers according to the version. If not, it prints out the long message from lines 2160-2220.

Lines 1720-1800 copy a JSR to my patch code over the top of the STA xxxx instruction which starts at either $2002 or $2008. It also modifies my patch code to include exactly the correct STA xxxx instruction which we are patching over. The patch code is at the very end of the Program Selector image, in lines 5970-6020. Later, when this patched ProDOS file is booted or otherwise executed, my patch code will install the boot drive ID into the ONLINE call block at line 5880.

Lines 1810-1950 copy the Program Selector image into the ProDOS image, at either $5700 or $5900 depending on version. Assuming we got this far, lines 1970-1990 will print out the "SUCCESSFUL" message.

```
                1000 *SAVE NEW.QUIT.CODE
                1010 *--------------------------------
                1020 *    Installation:
                1030 *      1.  BLOAD PRODOS,TSYS,A$2000
                1040 *      2.  BRUN INSTALL.QUITTER
                1050 *      3.  BSAVE PRODOS,TSYS,A$2000
                1051 *--------------------------------
                1052        .MA ASC           Macro to shorten listing
                1053        .AS -"]1"
                1054        .EM
                1060 *--------------------------------
                1070              .DUMMY
                1080              .OR 0000
0000-           1090 BPNTR        .BS 2
0002-           1100 SPNTR        .BS 2
0004-           1110 DPNTR        .BS 2
0006-           1120 DIR.INDEX    .BS 1
0007-           1130 DIR.START    .BS 1
0008-           1140 MAX.DIRPNT   .BS 1
0009-           1150 SEL.LINE     .BS 1
000A-           1160 MAX.LINE     .BS 1
000B-           1170 UNIT         .BS 1
000C-           1180 LENGTH       .BS 1
000D-           1190 CURTYP       .BS 1
000E-           1200 CURBLK       .BS 1
                1210 *--------------------------------
                1220              .OR $800
0800-           1230 OPNBUF       .BS 1024
0C00-           1240 DIRBUF       .BS 512
                1250              .ED
```

```
                        1260 *--------------------------------
0280-                   1270 PATHNAME   .EQ $280
2000-                   1280 BUFFER     .EQ $2000
2023-                   1290 ENTLEN     .EQ BUFFER+$23   ENTRY LENGTH
2024-                   1300 ENTCNT     .EQ BUFFER+$24   # ENTRIES PER BLOCK
                        1310 *--------------------------------
25-                     1320 CV      .EQ $25
32-                     1330 INVFLG  .EQ $32
                        1340 *--------------------------------
FB2F-                   1350 INIT       .EQ $FB2F
FC58-                   1360 HOME       .EQ $FC58
FC9C-                   1370 CLREOL     .EQ $FC9C
FDED-                   1380 COUT       .EQ $FDED
FD8E-                   1390 CROUT      .EQ $FD8E
FE80-                   1400 SETINV     .EQ $FE80
FE84-                   1410 SETNORM    .EQ $FE84
                        1420 *--------------------------------
BF00-                   1430 MLI     .EQ $BF00
BF58-                   1440 BITMAP  .EQ $BF58
                        1450 *--------------------------------
                        1460         .OR $1000
1000-                   1470         .TF INSTALL.QUITTER
                        1480 *--------------------------------
                        1490 INSTALL.QUITTER
1000- A2 57             1500         LDX /$5700   WHERE IMAGE IS IN 1.1.1
1002- A9 20             1510         LDA /$2000   WHERE LDA $43 IS IN 1.1.1
1004- 85 01             1520         STA BPNTR+1
1006- A0 00             1530         LDY #0
1008- 84 04             1540         STY DPNTR
100A- AD 00 20          1550         LDA $2000    SEE IF PRODOS IMAGE IS HERE
100D- C9 4C             1560         CMP #$4C     IF "JMP" THEN PROBABLY 1.4
100F- D0 04             1570         BNE .1       ...PROBABLY 1.1.1
1011- A0 06             1580         LDY #6       WHERE LDA $43 IS IN 1.4
1013- A2 59             1590         LDX /$5900   WHERE IMAGE IS IN 1.4
1015- 84 00             1600 .1      STY BPNTR    POINT AT LDA $43
1017- 86 05             1610         STX DPNTR+1  POINT AT IMAGE OF QUITTER
1019- E8                1620         INX
101A- E8                1630         INX
101B- 8E EC 13          1640         STX QPATCH+2 ADDRESS IN IMAGE OF "ONLINE+1"
101E- 8E F0 10          1650         STX JSR.QPATCH+2 ADDRESS IN IMAGE OF QPATCH
1021- A0 02             1660         LDY #2
1023- B9 EB 10          1670 .2      LDA PRODOS.ID.STRING,Y
1026- D1 00             1680         CMP (BPNTR),Y
1028- D0 34             1690         BNE .99      ...NEITHER, QUIT.
102A- 88                1700         DEY
102B- 10 F6             1710         BPL .2
                        1720 *---Trust we have ProDOS image---
102D- A0 02             1730         LDY #2       POINT AT STA XXXX
102F- B1 00             1740 .3      LDA (BPNTR),Y
1031- 99 EB 13          1750         STA QPATCH-2+3,Y
1034- B9 EC 10          1760         LDA JSR.QPATCH-2,Y
1037- 91 00             1770         STA (BPNTR),Y
1039- C8                1780         INY
103A- C0 05             1790         CPY #5
103C- 90 F1             1800         BCC .3
                        1810 *---Copy Quitter into ProDOS-----
103E- A9 F1             1820         LDA #QUIT.IMAGE
1040- 85 02             1830         STA SPNTR
1042- A9 10             1840         LDA /QUIT.IMAGE
1044- 85 03             1850         STA SPNTR+1
1046- A2 03             1860         LDX #3       COPY 3 PAGES
1048- A0 00             1870         LDY #0
104A- B1 02             1880 .4      LDA (SPNTR),Y
104C- 91 04             1890         STA (DPNTR),Y
104E- C8                1900         INY
104F- D0 F9             1910         BNE .4
1051- E6 03             1920         INC SPNTR+1
1053- E6 05             1930         INC DPNTR+1
1055- CA                1940         DEX
1056- D0 F2             1950         BNE .4
                        1960 *---Successful, say so and end---
1058- A0 00             1970         LDY #IQ.GOOD
105A- 20 68 10          1980         JSR IQ.PRINT
105D- 60                1990         RTS          FINISHED
                        2000 *---No ProDOS, or already patched---
105E- A0 27             2010 .99     LDY #IQ.BAD
1060- 20 68 10          2020         JSR IQ.PRINT
1063- 60                2030         RTS
                        2040 *--------------------------------
```

```
1064- 20 ED FD 2050 IQ.OUT JSR COUT
1067- C8       2060        INY
               2070 IQ.PRINT
1068- B9 6E 10 2080        LDA IQ,Y
106B- D0 F7    2090        BNE IQ.OUT
106D- 60       2100        RTS
               2110 *--------------------------------
106E-          2120 IQ     .EQ *
00-            2130 IQ.GOOD .EQ *-IQ
106E-          2140        >ASC "SUCCESSFULLY INSTALLED NEW QUIT CODE."
1093- 8D 00    2150        .HS 8D00
27-            2160 IQ.BAD .EQ *-IQ
1095-          2170        >ASC "EITHER PRODOS NOT HERE,"
10AC- 8D       2180        .HS 8D
10AD-          2190        >ASC "OR NOT VERSION 1.1.1 OR 1.4,"
10C9- 8D       2200        .HS 8D
10CA-          2210        >ASC "OR ALREADY INSTALLED QUIT CODE."
10E9- 8D 00    2220        .HS 8D00
               2230 *--------------------------------
               2240 PRODOS.ID.STRING
10EB- A5 43 8D 2250        .HS A5.43.8D
               2260 JSR.QPATCH
10EE- 20 F9 12 2270        JSR QPATCH.EP
               2280 *--------------------------------
               2290 QUIT.IMAGE
               2300        .PH $1000
               2310 *--------------------------------
               2320 QUITTER
1000- D8       2330        CLD             REQUIRED BY "STANDARDS"
1001- AD 82 C0 2340        LDA $C082       MOTHERBOARD ROMS
1004- 20 2F FB 2350        JSR INIT        TEXT MODE, FULL SCREEN WINDOW
1007- 20 84 FE 2360        JSR SETNORM
100A- A2 16    2370        LDX #$16
100C- A9 00    2380        LDA #0          PREPARE VIRGIN BITMAP
100E- 9D 58 BF 2390 .1     STA BITMAP,X
1011- 8E 6F BF 2400        STX BITMAP+$17    LAST TIME STORES $01, LOCK OUT $BF00 PAGE
1014- CA       2410        DEX
1015- D0 F7    2420        BNE .1
1017- A9 CF    2430        LDA #$CF
1019- 8D 58 BF 2440        STA BITMAP
               2450 *---LIST VOLUME NAMES------------
101C- A9 99    2460 .2     LDA #$99        CTRL-Y
101E- 20 00 C3 2470        JSR $C300       SET I/O HOOKS, 80-COL MODE, CLEAR SCREEN
1021- A0 00    2480        LDY #Q.SDV
1023- 20 68 12 2490        JSR MSG
1026- 20 DB 10 2500        JSR CLOSE.ALL.FILES
1029- A0 00    2510        LDY #0
102B- 84 08    2520        STY MAX.DIRPNT
102D- 84 07    2530        STY DIR.START
102F- 8C 80 02 2540        STY PATHNAME
1032- 8C 10 20 2550        STY BUFFER+16  (IN CASE "ONLINE" PRESET TO SPECIFIC S/D)
1035- 20 00 BF 2560        JSR MLI
1038- C5 E0 12 2570        .DA #$C5,ONLINE
103B- 84 09    2580 .3     STY SEL.LINE
103D- 20 32 11 2590        JSR DISPLAY.VOLUMES
1040- A0 11    2600        LDY #Q.VHELP
1042- 20 68 12 2610        JSR MSG
1045- 20 9D 11 2620        JSR GET.KEY
1048- 90 F1    2630        BCC .3          ...ARROW KEYS
104A- D0 D0    2640        BNE .2          ...ESCAPE KEY
               2650 *---READ DIRECTORY---------------
104C- 20 A5 10 2660 .4     JSR READ.THE.FILE
104F- B0 51    2670        BCS .7
               2680 *---PRINT PATHNAME---------------
1051- 20 58 FC 2690        JSR HOME
1054- A0 00    2700        LDY #0
1056- B9 81 02 2710 .5     LDA PATHNAME+1,Y
1059- 09 80    2720        ORA #$80
105B- 20 ED FD 2730        JSR COUT
105E- C8       2740        INY
105F- CC 80 02 2750        CPY PATHNAME
1062- 90 F2    2760        BCC .5
               2770 *---COLLECT FILENAMES------------
1064- A2 00    2780        LDX #0
1066- A9 FF    2790        LDA #$FF          FIRST JUST "SYS" FILES
1068- 20 E2 10 2800        JSR SCAN.DIRECTORY
106B- A9 0F    2810        LDA #$0F          THEN JUST "DIR" FILES
106D- 20 E2 10 2820        JSR SCAN.DIRECTORY
1070- 8A       2830        TXA             SEE IF ANY FILES FOUND
1071- F0 A9    2840        BEQ .2          ...NO, BACK TO THE TOP
1073- A9 00    2850        LDA #0          MARK END OF LIST
1075- 9D 00 0D 2860        STA DIRBUF+256,X
1078- 86 08    2870        STX MAX.DIRPNT
```

```
                        2880 *---LIST THE FILENAMES-----------
107A- A8                2890         TAY             Y=0
107B- 84 07             2900         STY DIR.START
107D- 84 09             2910 .6      STY SEL.LINE
107F- 20 01 12          2920         JSR DISPLAY.FILES
1082- A0 11             2930         LDY #Q.VHELP
1084- 20 68 12          2940         JSR MSG
1087- 20 9D 11          2950         JSR GET.KEY
108A- 90 F1             2960         BCC .6          ...ARROW KEYS
108C- D0 8E             2970         BNE .2          ...ESCAPE KEY
108E- A0 10             2980         LDY #$10
1090- B1 02             2990         LDA (SPNTR),Y      GET FILE TYPE
1092- 10 B8             3000         BPL .4             DIRECTORY ($0F)
                        3010 *---SYS FILE, LOAD & EXECUTE-----
1094- 20 00 BF          3020         JSR MLI         SET PREFIX
1097- C6 F2 12          3030         .DA #$C6,PATH
109A- 20 A5 10          3040         JSR READ.THE.FILE
109D- B0 03             3050         BCS .7          ...ERROR IN READING
109F- 4C 00 20          3060         JMP BUFFER
10A2- 4C 00 10          3070 .7      JMP QUITTER
                        3080 *-------------------------------
                        3090 READ.THE.FILE
10A5- A0 00             3100         LDY #0          APPEND CURRENTLY SELECTED NAME
10A7- B1 02             3110         LDA (SPNTR),Y      GET LENGTH OF NAME
10A9- 29 0F             3120         AND #$0F
10AB- 85 0C             3130         STA LENGTH
10AD- AE 80 02          3140         LDX PATHNAME       CURRENT LENGTH
10B0- A9 2F             3150         LDA #'/'
10B2- E8                3160 .1      INX
10B3- C8                3170         INY
10B4- 9D 80 02          3180         STA PATHNAME,X
10B7- B1 02             3190         LDA (SPNTR),Y
10B9- C6 0C             3200         DEC LENGTH
10BB- 10 F5             3210         BPL .1
10BD- 8E 80 02          3220         STX PATHNAME
10C0- 20 00 BF          3230         JSR MLI            OPEN THE FILE
10C3- C8 E4 12          3240         .DA #$C8,OPEN
10C6- B0 19             3250         BCS RF.ERR
10C8- AD E9 12          3260         LDA O.REF          FILE REFERENCE NUMBER
10CB- 8D EB 12          3270         STA R.REF
10CE- 20 00 BF          3280         JSR MLI            READ THE WHOLE FILE
10D1- CA EA 12          3290         .DA #$CA,READ
10D4- 90 05             3300         BCC CLOSE.ALL.FILES
10D6- C9 4C             3310         CMP #$4C           IS IT JUST EOF?
10D8- 38                3320         SEC
10D9- D0 06             3330         BNE RF.ERR         ...NO
                        3340 CLOSE.ALL.FILES
10DB- 20 00 BF          3350         JSR MLI            CLOSE THE FILE
10DE- CC DE 12          3360         .DA #$CC,CLOSE
10E1- 60                3370 RF.ERR RTS
                        3380 *-------------------------------
                        3390 SCAN.DIRECTORY
10E2- 85 0D             3400         STA CURTYP         TYPE WE ARE COLLECTING
10E4- A9 00             3410         LDA #0             START WITH FIRST BLOCK
10E6- 85 0E             3420 .1      STA CURBLK
10E8- A9 04             3430         LDA #BUFFER+4      FIRST 4 BYTES OF BLOCK SKIPPED
10EA- 85 04             3440         STA DPNTR
10EC- 18                3450         CLC                COMPUTE PAGE OF PNTR
10ED- A9 20             3460         LDA /BUFFER+4
10EF- 65 0E             3470         ADC CURBLK
10F1- 85 05             3480         STA DPNTR+1
10F3- AD 24 20          3490         LDA ENTCNT
10F6- 85 0C             3500         STA LENGTH
                        3510 *-------------------------------
10F8- A0 00             3520 .2      LDY #0
10FA- B1 04             3530         LDA (DPNTR),Y
10FC- 29 F0             3540         AND #$F0
10FE- F0 17             3550         BEQ .4          ...DELETED FILE
1100- C9 E0             3560         CMP #$E0        ...HEADER?
1102- B0 13             3570         BCS .4          ...YES
1104- A0 10             3580         LDY #$10
1106- B1 04             3590         LDA (DPNTR),Y      LOOK AT FILE TYPE
1108- C5 0D             3600         CMP CURTYP
110A- D0 0B             3610         BNE .4             ...NOT CURRENT TYPE
                        3620 *---DIR or SYS file--------------
110C- A5 04             3630 .3      LDA DPNTR
110E- 9D 00 0C          3640         STA DIRBUF,X
1111- A5 05             3650         LDA DPNTR+1
1113- 9D 00 0D          3660         STA DIRBUF+256,X
1116- E8                3670         INX
```

```
                     3680 *---ADVANCE TO NEXT ENTRY--------
1117- 18             3690 .4     CLC
1118- A5 04          3700        LDA DPNTR
111A- 6D 23 20       3710        ADC ENTLEN
111D- 85 04          3720        STA DPNTR
111F- 90 02          3730        BCC .5
1121- E6 05          3740        INC DPNTR+1
1123- C6 0C          3750 .5     DEC LENGTH          AT END OF BLOCK YET?
1125- D0 D1          3760        BNE .2              ...NO, CONTINUE IN BLOCK
1127- 18             3770        CLC
1128- A5 0E          3780        LDA CURBLK
112A- 69 02          3790        ADC #2
112C- CD F1 12       3800        CMP ACTLEN+1
112F- 90 B5          3810        BCC .1              ...YES, READ NEXT BLOCK
1131- 60             3820        RTS
                     3830 *-------------------------------
                     3840 DISPLAY.VOLUMES
1132- 20 43 12       3850        JSR SETUP.DISPLAY.LOOP
1135- A9 20          3860        LDA /BUFFER
1137- 85 01          3870        STA BPNTR+1
1139- A9 00          3880        LDA #BUFFER
113B- 85 00          3890 .1     STA BPNTR
113D- A0 00          3900        LDY #0
113F- B1 00          3910        LDA (BPNTR),Y
1141- F0 35          3920        BEQ .5              ...END OF LIST
1143- 29 0F          3930        AND #$0F
1145- F0 2A          3940        BEQ .3        ...NO VOLUME HERE
                     3950 *-------------------------------
1147- 20 51 12       3960        JSR CHECK.FOR.SEL.LINE
                     3970 *-------------------------------
114A- B1 00          3980        LDA (BPNTR),Y       GET UNIT NUMBER
114C- 4A             3990        LSR                 ISOLATE SLOT NUMBER
114D- 4A             4000        LSR
114E- 4A             4010        LSR
114F- 4A             4020        LSR
1150- 29 07          4030        AND #7
1152- 09 B0          4040        ORA #"0"
1154- 20 ED FD       4050        JSR COUT            PRINT SLOT NUMBER
1157- A9 AF          4060        LDA #"/"
1159- 20 ED FD       4070        JSR COUT
115C- B1 00          4080        LDA (BPNTR),Y       GET UNIT NUMBER AGAIN
115E- 0A             4090        ASL                 SET CARRY IF DRIVE 2
115F- A9 B1          4100        LDA #"1"            ASSUME DRIVE 1
1161- 69 00          4110        ADC #0              CHANGE TO 2 IF TRUE
1163- 20 ED FD       4120        JSR COUT
1166- A9 A0          4130        LDA #" "      PRINT TWO SPACES
1168- 20 ED FD       4140        JSR COUT
116B- 20 ED FD       4150        JSR COUT
116E- 20 79 11       4160        JSR PRINT.BPNTR.NAME
                     4170 *-------------------------------
1171- 18             4180 .3     CLC                 POINT TO NEXT VOLUME NAME
1172- A5 00          4190        LDA BPNTR
1174- 69 10          4200        ADC #16
1176- 90 C3          4210        BCC .1        STILL IN SAME PAGE
1178- 60             4220 .5     RTS
                     4230 *-------------------------------
                     4240 PRINT.BPNTR.NAME
1179- A0 00          4250        LDY #0
117B- B1 00          4260        LDA (BPNTR),Y       GET NAME LENGTH
117D- 29 0F          4270        AND #$0F
117F- AA             4280        TAX
1180- C8             4290 .1     INY           PRINT THE VOLUME OR FILE NAME
1181- B1 00          4300        LDA (BPNTR),Y
1183- 09 80          4310        ORA #$80
1185- 20 ED FD       4320        JSR COUT
1188- CA             4330        DEX
1189- D0 F5          4340        BNE .1
                     4350 *-------------------------------
118B- A9 A0          4360 .2     LDA #" "      PRINT TRAILING BLANKS
118D- 20 ED FD       4370        JSR COUT
1190- C8             4380        INY
1191- C0 10          4390        CPY #16
1193- 90 F6          4400        BCC .2
1195- 20 84 FE       4410        JSR SETNORM  NORMAL MODE NOW
1198- E6 0A          4420        INC MAX.LINE        COUNT THE LINE
119A- 4C 8E FD       4430        JMP CROUT
                     4440 *-------------------------------
```

```
                        4450 GET.KEY
119D- A4 09             4460         LDY SEL.LINE       CURRENT BRIGHT LINE
119F- AD 00 C0          4470 .1      LDA $C000     READ KEY FROM KEYBOARD
11A2- 10 FB             4480         BPL .1
11A4- 8D 10 C0          4490         STA $C010     CLEAR THE STROBE
11A7- C9 B0             4500         CMP #$B0      CHECK FOR "0"..."7"
11A9- 90 0F             4510         BCC .12
11AB- C9 B8             4520         CMP #$B8
11AD- F0 25             4530         BEQ .25
11AF- B0 EE             4540         BCS .1
11B1- 0A                4550         ASL
11B2- 0A                4560         ASL
11B3- 0A                4570         ASL
11B4- 0A                4580         ASL           LEAVE SLOT*16 IN A, CARRY SET
11B5- 8D E1 12          4590 .11     STA ONLINE+1  CHANGE ONLINE CALL
11B8- A9 9B             4600         LDA #$9B      SIMULATE AN <ESCAPE>
11BA- C9 8D             4610 .12     CMP #$8D
11BC- F0 15             4620         BEQ .2        <RETURN>
11BE- C9 88             4630         CMP #$88      <--
11C0- F0 19             4640         BEQ .3
11C2- C9 95             4650         CMP #$95      -->
11C4- F0 26             4660         BEQ .7
11C6- C9 8A             4670         CMP #$8A      DOWN ARROW
11C8- F0 22             4680         BEQ .7
11CA- C9 8B             4690         CMP #$8B      UP ARROW
11CC- F0 0D             4700         BEQ .3
11CE- C9 9B             4710         CMP #$9B      ESCAPE
11D0- D0 CD             4720         BNE .1        GET ANOTHER CHARACTER
11D2- 0A                4730         ASL           ...SET .NE. and CARRY
11D3- 60                4740 .2      RTS
11D4- AD E1 12          4750 .25     LDA ONLINE+1
11D7- 09 80             4760         ORA #$80           TRY DRIVE 2
11D9- D0 DA             4770         BNE .11            ...ALWAYS
                        4780 *---<UP OR LEFT ARROW>-----------
11DB- 18                4790 .3      CLC
11DC- 88                4800         DEY                IS CURRENT BRIGHT LINE TOP LINE?
11DD- 10 21             4810         BPL .8             ...NOT TOP LINE
11DF- A4 07             4820         LDY DIR.START      ARE WE DISPLAYING THE FIRST ONE?
11E1- F0 05             4830         BEQ .5             ...YES
11E3- C6 07             4840         DEC DIR.START      ...NO, MOVE TOWARD FIRST LINE
11E5- A0 00             4850 .4      LDY #0             MAKE FIRST LINE BRIGHT
11E7- 60                4860         RTS
11E8- A4 0A             4870 .5      LDY MAX.LINE       MAKE LAST LINE BRIGHT
11EA- 88                4880         DEY
11EB- 60                4890         RTS
                        4900 *---<DOWN OR RIGHT ARROW>--------
11EC- C8                4910 .7      INY                MOVE TOWARD LAST LINE
11ED- C4 0A             4920         CPY MAX.LINE       BEYOND END OF SCREEN?
11EF- 90 0F             4930         BCC .8             ...NO
11F1- A5 08             4940         LDA MAX.DIRPNT     ...YES, CHECK IF SHOWING LAST LINE
11F3- E9 11             4950         SBC #17
11F5- 90 EE             4960         BCC .4             ...YES
11F7- C5 07             4970         CMP DIR.START
11F9- 90 EA             4980         BCC .4             ...YES
11FB- E6 07             4990         INC DIR.START      ...NO, MOVE TOWARD LAST LINE
11FD- A4 09             5000         LDY SEL.LINE
11FF- 18                5010         CLC
1200- 60                5020 .8      RTS
                        5030 *-------------------------------
                        5040 DISPLAY.FILES
1201- 20 43 12          5050         JSR SETUP.DISPLAY.LOOP
1204- A5 07             5060         LDA DIR.START
1206- 85 06             5070         STA DIR.INDEX
1208- 20 37 12          5080         JSR CLEAR.LINE.OR.PRINT.MORE.MSG
                        5090 *-------------------------------
120B- A6 06             5100 .1      LDX DIR.INDEX
120D- BC 00 0D          5110         LDY DIRBUF+256,X
1210- F0 21             5120         BEQ .4             ...END OF LIST
1212- 84 01             5130         STY BPNTR+1
1214- BD 00 0C          5140         LDA DIRBUF,X
1217- 85 00             5150         STA BPNTR
1219- 20 51 12          5160         JSR CHECK.FOR.SEL.LINE
                        5170 *-------------------------------
121C- A0 10             5180 .2      LDY #$10
121E- B1 00             5190         LDA (BPNTR),Y
1220- 30 03             5200         BMI .3             ...SYS FILE
1222- A0 5C             5210         LDY #Q.DIR
1224- 2C                5220         .HS 2C
1225- A0 54             5230 .3      LDY #Q.SYS
1227- 20 68 12          5240         JSR MSG
122A- 20 79 11          5250         JSR PRINT.BPNTR.NAME
```

```
                    5260 *--------------------------------
122D- E6 06         5270         INC DIR.INDEX
122F- C6 0C         5280         DEC LENGTH
1231- D0 D8         5290         BNE .1
1233- A5 06         5300 .4      LDA DIR.INDEX
1235- C5 08         5310         CMP MAX.DIRPNT
                    5320 *--------------------------------
                    5330 CLEAR.LINE.OR.PRINT.MORE.MSG
1237- F0 04         5340         BEQ .1          CLEAR LINE
1239- A0 64         5350         LDY #Q.MORE
123B- D0 2B         5360         BNE MSG         ...ALWAYS
123D- 20 9C FC      5370 .1      JSR CLREOL
1240- 4C 8E FD      5380         JMP CROUT
                    5390 *--------------------------------
                    5400 SETUP.DISPLAY.LOOP
1243- A9 10         5410         LDA #16         MAX 16 LINES IN LIST
1245- 85 0C         5420         STA LENGTH
1247- A0 00         5430         LDY #0
1249- 84 0A         5440         STY MAX.LINE
124B- C8            5450         INY             SAME AS VTAB 3, HTAB 1
124C- 84 25         5460         STY CV
124E- 4C 8E FD      5470         JMP CROUT
                    5480 *--------------------------------
                    5490 CHECK.FOR.SEL.LINE
1251- A5 0A         5500         LDA MAX.LINE       SEE IF CURRENT LINE SHOULD
1253- C5 09         5510         CMP SEL.LINE            BE INVERSE MODE
1255- D0 0C         5520         BNE .1             :::NO
1257- A5 00         5530         LDA BPNTR          :::YES, SO SETUP POINTER
1259- 85 02         5540         STA SPNTR
125B- A5 01         5550         LDA BPNTR+1
125D- 85 03         5560         STA SPNTR+1
125F- A9 3F         5570         LDA #$3F                  & SET INVERSE MODE
1261- 85 32         5580         STA INVFLG
1263- 60            5590 .1      RTS
                    5600 *--------------------------------
1264- 20 ED FD      5610 MSG1    JSR COUT
1267- C8            5620         INY
1268- B9 6E 12      5630 MSG     LDA QTS,Y
126B- D0 F7         5640         BNE MSG1
126D- 60            5650         RTS
                    5660 *--------------------------------
126E-               5670 QTS     .EQ *
00-                 5680 Q.SDV   .EQ *-QTS
126E-               5690         >ASC "S/D  Volume Name"
127E- 00            5700         .HS 00
11-                 5710 Q.VHELP .EQ *-QTS
127F- 8D            5720         .HS 8D
1280-               5730         >ASC "Select with ARROWS and <RETURN>"
129F- 8D            5740         .HS 8D
12A0-               5750         >ASC "See Volumes with <ESCAPE> or 0-8"
12C0- 8D 00         5760         .HS 8D00
54-                 5770 Q.SYS   .EQ *-QTS
12C2-               5780         >ASC "SYS -- "
12C9- 00            5790         .HS 00
5C-                 5800 Q.DIR   .EQ *-QTS
12CA-               5810         >ASC "DIR -- "
12D1- 00            5820         .HS 00
64-                 5830 Q.MORE  .EQ *-QTS
12D2-               5840         >ASC "<<<MORE>>>"
12DC- 8D 00         5850         .HS 8D00
                    5860 *--------------------------------
12DE- 01 00         5870 CLOSE   .DA #1,#0
12E0- 02 00 00
12E3- 20            5880 ONLINE  .DA #2,#0,BUFFER
12E4- 03 80 02
12E7- 00 08         5890 OPEN    .DA #3,PATHNAME,OPNBUF
12E9-               5900 O.REF   .BS 1
12EA- 04            5910 READ    .DA #4
12EB-               5920 R.REF   .BS 1
12EC- 00 20 00
12EF- 9F            5930         .DA BUFFER,$9F00
12F0-               5940 ACTLEN  .BS 2
12F2- 01 80 02      5950 PATH    .DA #1,PATHNAME
                    5960 *--------------------------------
12F5-               5970         .BS $1300-*-7
                    5980 QPATCH.EP
                    5990         .EP
13EA- 8D E1 12      6000 QPATCH  STA ONLINE+1
13ED- 8D ED 13      6010         STA *
13F0- 60            6020         RTS
                    6030 *--------------------------------
```

Special Version of S-C Macro for Huge Symbol Tables

Sometimes there just is not enough memory to hold the entire symbol table of a large assembly, especially in the ProDOS version of the S-C Macro Assembler. In that version, the symbol table normally begins at $1000 and grows upward, while the source program snugs up against $7400, hanging downward from there. Even with the use of the .INB directive to bring in segments of the source code one disk block at a time, extra-large programs can run out of memory during assembly. It can be especially frustrating in an Apple //e or later machine, when you know there is a lot of free memory just across the Soft-Switch River, over there in Aux-land.

Until now I have resisted using this memory, trying to remain fully compatible with older 64K machines and trying to keep the speed advantages of all-main-memory. But finally, the need became personal enough. I created a special version which puts the entire symbol table in Aux RAM. It will run on a 128K or larger //e, //c, or IIgs; I haven't tried it, but it should also run in an Apple II Plus with an Applied Engineering Transwarp card plugged and turned on. The symbol table still begins at $1000, but in AuxRAM; and it can rise as high as $BFFF without challenge. That is $B000 total bytes, or about twice as many as were available between $1000 and the bottom of the source program in MainRAM. The space below $1000, down as far as $800, is occupied by macro private labels, if you use any. Obviously, there is also now more room in Main RAM for your source code and/or object code. A Minus: if you have a /RAM disk installed in AuxRAM, the symbol table walks all over it. However, if you are using a RamWorks card or the like, with their PRODRIVE software, bank 0 of AuxRAM is left available for just these type uses.

If you need something like this, it's finally here. I'm calling it Version 2.1, and as a registered owner of the ProDOS version of the S-C Macro Assembler you can have a copy for only $10.

"IIgs Toolbox Reference" Now Available

At long last, Addison-Wesley assures me they have printed the "IIgs Toolbox Reference" manuals, Volumes 1 and 2. These are probably indispensable tools for any serious IIgs programmers. Until now, you could only get them in beta versions through APDA. They are over 750 pages each, and cost $26.95 each ($24 plus shipping charge if you order from us). Each of the standard tools is described in great detail, with examples in both assembly language and C.

If they kept the same arrangement as my pre-beta copy (dated Nov 1986), there are a total of 23 chapters in the set. In my edition, Volume 1 covers toolsets in general, Desktop Bus, Control Mngr, Desk Mngr, Dialog Mngr, Event Mngr, Font Mngr, Intger Math, Line Edit, Memory Mngr, Menu Mngr, Misc.Tools, and Print Mngr; Volume 2 covers Quickdraw, SANE, Scheduler, Scrap Mngr, Sound Mngr, Std. File Operations, Text.tools, Tool Locater, and Window Manager. Volume 2 also includes an appendix on writing your own tool set.

There remains one more book in the IIgs series, the "IIgs Programmer's Introduction". A-W is now predicting April '88 for the publishing date, and a price of $32.95. I guess I am old-fashioned, but to my mind this book should have been published FIRST, and included FREE with every IIgs purchase. Oh, well.... We will accept orders on this one now, and hold them until the book comes, at a price of $30 plus shipping (see note on page 3 for details on shipping charges).

Apple Assembly Line (ISSN 0889-4302) is published monthly by S-C SOFTWARE CORPORATION, P.O. Box 280300, Dallas, Texas 75228. Phone (214) 324-2050. Subscription rate is $24 per year in the USA, Canada, and Mexico, or $36 in other countries. Back issues are $1.80 each for Volumes 1-7, $2.40 each from later Volumes (plus postage). A subscription to the newsletter with a Monthly Disk containing all program source code and article text is $64 per year in the USA, Canada and Mexico, and $90 to other countries.